.

Me. Bruno Klaus de Aquino Afonso

`LGC_LVO_Auto` : Correction of Labels with Gradient Descent Optimization for Graph-based Semi-supervised Learning

São José dos Campos

Me. Bruno Klaus de Aquino Afonso

`LGC_LVO_Auto` : Correction of Labels with Gradient Descent Optimization for Graph-based Semi-supervised Learning

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal de São Paulo – UNIFESP

Instituto de Ciência e Tecnologia

Bacharelado em Ciência da Computação

Supervisor: Profª. Drª. Lilian Berton

São José dos Campos

2020

Elaborado por sistema de geração automática com os dados fornecidos pelo(a) autor(a).

de Aquino Afonso, Bruno Klaus
LGC_LVO_AUTO: Correction of Labels with Gradient Descent Optimization for Graph-based Semi-supervised Learning/ Bruno Klaus de Aquino Afonso Orientador(a) Lilian Berton-São José dos Campos, 2020.
89 p.

Trabalho de Conclusão de Curso-Bacharelado em Ciência da Computação-Universidade Federal de São Paulo-Instituto de Ciência e Tecnologia, 2020.

1. Aprendizado de Máquina. 2. Ruído de Rótulo. 3. Aprendizado Semissupervisionado baseado em Grafo. 4. Filtro. 5. Consistência Local e Global. I. Berton, Lilian, orientador(a). II. Título.

Este trabalho é dedicado a todos pesquisadores que tiveram que superar dificuldades profissionais ou pessoais no ano de 2020.

Acknowledgements

I would like to thank Lilian Berton for diligently supervising this work.

I would also like to thank my family for the emotional support.

Lastly, I thank everyone in UNIFESP's (Federal University of São Paulo) Institute of Science and Tecnology or elsewhere that has had to deal with and adapt to a new reality during 2020.

"In practice, experienced machine-learning engineers and researchers build intuition over time as to what works and what doesn't when it comes to these choices—they develop hyperparameter-tuning skills. But there are no formal rules. If you want to get to the very limit of what can be achieved on a given task, you can't be content with arbitrary choices made by a fallible human. Your initial decisions are almost always suboptimal, even if you have good intuition. You can refine your choices by tweaking them by hand and retraining the model repeatedly—that's what machine-learning engineers and researchers spend most of their time doing. But it shouldn't be your job as a human to fiddle with hyperparameters all day—that is better left to a machine."

(François Chollet)

Resumo

Nesse trabalho, consideramos o problema de aprendizado de máquina com rótulos ruidosos, no caso em que a maioria dos dados não são rotulados. Mais especificamente, nos concentramos na aprendizagem semissupervisionada baseada em grafos, em que muitas abordagens diferentes já foram propostas, como a norma ℓ_1, a busca de base de autofunções suave e a formulação bivariada. Propomos nosso próprio filtro semissupervisionado, nomeado **Filtro automático *leave-one-out* com base na consistência local e global (`LGC_LVO_Auto`)**, que corrige e redistribui as informações do rótulo para minimizar o erro *leave-one-out* (deixa um rótulo de fora), ao mesmo tempo mantendo-se consistente com o processo de passeio aleatório imposto por sua linha de base, o algoritmo de Consistência Local e Global (LGC). Exploramos o problema da dominância diagonal em soluções do LGC e sua possível relação com o sobreajuste, e como zerando essa diagonal leva ao custo desejado. Fazemos uso da otimização via gradiente descendente nos rótulos para minimizar esse custo, transferindo parte da confiança nos próprios rótulos para o modelo de propagação. Para eliminar soluções degeneradas, algumas restrições são postas em prática: os rótulos não podem mudar de classe e a contribuição geral de cada classe deve permanecer a mesma. A otimização requer apenas as relações entre os rótulos: consequentemente, é adequado para conjuntos de dados moderadamente grandes, em particular quando os dados rotulados são escassos. Requer um único parâmetro. Em teoria, ele pode ser estendido trivialmente para uma generalização de sua linha de base. Os resultados mostram que `LGC_LVO_Auto` é capaz de superar sua linha de base com sobra quando há ruído e atrapalha pouco no cenário sem ruído, sendo uma ferramenta útil para a detecção de rótulos ruidosos. Além disso, é competitivo com outros métodos que requerem mais parâmetros.

Palavras-chaves: Aprendizado de Máquina; Ruído de Rótulo; Aprendizado Semissupervisionado baseado em Grafos; Filtro; Consistência Local e Global.

Abstract

We consider the problem of learning with noisy labels where most of the data is unlabelled. More specifically, we focus on graph-based semi-supervised learning, a setting in which many different approaches have already been proposed, such as ℓ_1 norm, smooth eigenbasis pursuit and bivariate formulation. We propose our own semi-supervised filter, named **Automatic Leave-One-Out Filter based on Local and Global Consistency** (`LGC_LVO_Auto`), that corrects and redistributes label information in order to minimize leave-one-out error, while remaining consistent with the random walk process imposed by its baseline, the Local and Global Consistency (LGC) algorithm. We explore the problem of diagonal dominance in LGC solutions and its possible relation to overfitting, and how setting it to zero leads to the leave-one-out cost. We make use of gradient descent optimization on labels to minimize this cost, transferring some of the trust from the labels themselves to the propagation model. In order to eliminate degenerate solutions, some restrictions are put in place: labels cannot change class, and the overall contribution for each class should remain the same. The optimization requires only the relations between labels: consequently, it is suited to moderately large datasets such as MNIST, in particular when labelled data is scarce. It requires a single parameter. In theory, it may be extended trivially to the more general LapRLS classifier. Results show that `LGC_LVO_Auto` is capable of outperforming its baseline significantly when there is noise, and not be too harmful in the noiseless scenario. Moreover, it is competitive with other methods that require more parameters.

Keywords: Machine Learning; Noisy labels; Graph-Based Semi-Supervised Learning;Filter; Local and Global Consistency.

List of Figures

List of Tables

List of abbreviations and acronyms

GFHF	Gaussian Fields and Harmonic Functions
GSSL	Graph-Based Semi-Supervised Learning
GTAM	Graph Transduction Based on Alternating Minimization
GTF	Graph Trend Filtering
KNN	k-Nearest Neighbours
LDST	Label Diagnosis Through Self-Tuning
LE	Laplacian Eigenmaps
LGC	Local and Global Consistency
`LGC_LVO`	Leave-One-Out Approach based on Local and Global Consistency
`LGC_LVOf`	Leave-One-Out Filter based on Local and Global Consistency
`LGC_LVO_Auto`	Automatic Leave-One-Out filter based on Local and Global Consistency
LOO	Leave-One-Out
LSSC	Large-Scale Sparse Coding
ML	Machine Learning
RBF	Radial Basis Function
SPMR	Self-Paced Manifold Regularization
SSL	Semi-Supervised Learning

List of symbols

n	Number of instances
d	Number of dimensions
c	Number of classes
l	Number of labels
$\mathbf{X}$	Input matrix (size $n \times d$)
$\mathbf{Y}$	(True) label matrix (size $n \times c$)
$\mathbf{F}$	Classification matrix (size $n \times c$)
$\mathbf{W}$	Weight matrix (diagonal with n entries)
$\widetilde{W}$	Transition probability matrix (size $n \times n$)
$\mathbf{D}$	Degree matrix (diagonal with n entries)
$\mathbf{I}$	Identity matrix (diagonal with n entries)
$\mathbf{L}_{\mathbb{C}}$	Combinatorial graph Laplacian (size $n \times n$)
$\mathbf{L}_{\mathbb{N}}$	Normalized graph Laplacian (size $n \times n$)
$\mathbf{L}$	Arbitrary or unspecified graph Laplacian (size $n \times n$)
$\mathbb{R}^{i \times j}$	Vector space of matrices of size $i \times j$
Λ_m	Eigenvalue matrix (diagonal,with m entries)
$\mathbf{K}$	Kernel matrix (size $n \times n$)
Σ	Regularization matrix (size $n \times n$)
$\widetilde{\mathbf{Y}}$	Normalized label matrix (size $n \times c$)
$\mathbf{\Omega}$	Optimized coefficients for label consistency (diagonal, l entries)

Contents

1 Introduction

The term *Machine Learning* has been around for many decades, being already present in the 1950s (SAMUEL, 1959). Yet, there is no doubt that this area of research has been through a surge of popularity recently (AFONSO et al., 2019; CHEN, 2016; YOUNG et al., 2018). Why is that so? The often repeated, but always relevant definition by Mitchell (1997) tells us that:

> "A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**".

In general, a performance measure is only there to serve its purpose, and the tasks we have successfully tackled most certainly increased in difficulty as the field has gotten more mature. This implies one of two things: either we developed novel methods that make the most of the experiences we previously had, or we are now able to provide better experiences for our computer program. As it turns out, both things go hand-in-hand. Much of the important development, particularly in Deep Learning research (LECUN; BENGIO; HINTON, 2015), has enabled unforeseeable breakthroughs, with a caveat: it needs both the power of modern hardware, and the kind of experience which could rarely be feasible until now.

At the core of the experience, is **data**. Complex models are expected to require more data to be properly fine-tuned. As such, it is no surprise to see buzzwords such as "big data" emphasized. This one in particular, it combines well with the mass adoption of portable devices such as smartphones, which means that there is a continuous stream of data to be collected. This is pushed even further when one takes into account devices that make up the Internet of things (IoT) (RIGGINS; WAMBA, 2015), ranging from electronic appliances to vending machines (SOLANO et al., 2017).

According to LeCun, Bengio e Hinton (2015), the increase in model size is a consequence of GPU development, faster network connectivity and infrastructure for distributed computing, and has had a noticeable impact on tackling many different and difficult tasks. In spite of this, it is not enough to have a huge volume of data. The other side of big data is *data quality* (SAHA; SRIVASTAVA, 2014). Machine learning approaches are divided by *supervision*. Suppose that we have countless images of a drone hovering over a given city. We could subdivide this image into patches and ask, for example, if it contains a sidewalk. This output variable, which is to be interpreted as the desired output of our algorithm, is commonly called a **label**. When labels are plenty, the input-output pairs are enough for training a complex and high-performing model. This is called *supervised learning* (ALPAYDIN, 2010). Crucially, the labelling process is often something that has to be done manually, which would be impractical

to do for hours of recorded drone footage. The main takeaway here is that obtaining labelled data is usually more difficult and costly to obtain compared to unlabelled data.

Collecting large unlabelled data is useless if we don't know what to do with it. Unlabelled data can be useful by itself, as many insights may be obtained via *unsupervised learning*, such as cluster/network analysis and all kinds of exploratory data visualization techniques (SOUSA; DEL-FIACO; BERTON, 2018). However, the end goal here is to use unlabelled data to fine-tune our label-predicting classifier. This is the objective that *semi-supervised learning* (SSL) sets out to accomplish. This learning paradigm uses unlabelled data to improve a solution by restricting the hypothesis space, or adding extra regularization (CHAPELLE; SCHÖLKOPF; ZIEN, 2006). When datasets are consistent with the assumptions, the gap between a semi-supervised classifier and one trained with a fully labelled dataset is closed significantly, while expenses are kept to a minimum.

It is reasonable to wonder how the unlabelled data can possibly have information that helps us determine the label. In truth, semi-supervised learning does depend more on assumptions than its supervised counterpart. According to a 2020 survey on semi-supervised learning (ENGELEN; HOOS, 2020), one of the most important issues to be addressed is the potential performance degradation caused by the introduction of unlabelled data. "Safe" semi-supervised learning is the idea of never doing worse than the supervised baseline for any labelling of the unlabelled data. This is explored by Krijthe (2018), which includes the development of a safe semi-supervised least squares classifier. On the other hand, a mathematical proof is given that safeness is unattainable for linear classifiers defined by convex margin-based surrogate losses that are monotonically decreasing (KRIJTHE, 2018).

The fear of performance degradation definitely has an effect on the willingness of the average practitioner to experiment with semi-supervised learning. According to Engelen e Hoos (2020), one notable exception is the recent advances in deep semi-supervised learning, attributed to the ease of adapting the loss function to include unlabelled data, and the relatively weak *smoothness assumption*: minor variations in the input space should only cause minor variations in the output space. *Graph-Based Semi-Supervised learning* (GSSL) (ZHU, 2005a), which is central to our work, is a slightly older method that uses a similar smoothness assumption. Although it has some computational issues when dealing with huge datasets, there are distinct advantages: we can combine labelled and unlabelled data in a principled way, making use of the intrinsic geometric nature of the data. As a result, GSSL has been used successfully in a wide variety of applications, such as drug-protein interaction prediction (XIA et al., 2010), sentiment analysis (HAMILTON et al., 2016), word sense disambiguation (YUAN et al., 2016; SOUSA; MILIOS; BERTON, 2020) and character recognition (CATUNDA; SILVA; BERTON, 2019). Another benefit is that most of GSSL can be expressed clearly using matrix algebra, which makes it more manageable to tackle the other problem we might face: our few labels may also not be wholly reliable.

Semi-supervised algorithms were initially designed to, first and foremost, compensate for the few available labels. It would, therefore, make sense that the semi-supervised classifier should fully trust these labels. However, this line of thought falls apart quickly when you consider real-world datasets. More specifically, one can look at any crowdsourcing approach such as the Amazon Mechanical Turk (CROWSTON, 2012), which has been shown to be a very efficient tool for obtaining labels. According to Chang, Amershi e Kamar (2017), researchers have reported difficulty obtaining high quality labels through crowdsourcing due to a few factors: inattentive labellers, uncertainty in the task itself, and varying worker knowledge. To make a SSL method robust, the labelling process should be assumed to be imperfect. In other words, the labels we are working with are said to be *weak*, or *noisy*. The true label is not actually observed, instead it may be modelled as a latent variable, whereas the observed label is the result of subjecting it to a *noise process*.

Noise can be defined as anything obscuring the relationship between the class and the features (HICKEY, 1996), i.e. the input and output. We are concerned with label noise (also known as class noise), which happens whenever labels are corrupted by some underlying noise model, making it so that they cannot be fully trusted. There are three ways of dealing with label noise. First, one could use *label noise-robust models* that do not explicitly handle label noise. These can only hope that the usual overfitting avoidance mechanisms will also be useful to lessen the impact of label noise. This may be a consequence, e.g., of the chosen loss function. It has been shown that the squared error loss function is tolerant only to uniform noise (MANWANI; SASTRY, 2013). The second type of approach would be the use of *filters* to eliminate the noise from the training set before using it for learning a classifier. This can be done by either removing instances with noisy labels or trying to correct them (TENG, 2015). Thirdly, *label noise-tolerant* methods do consider label noise directly. In particular, supervised and semi-supervised learning algorithms can be modified to be tolerant to noise. A comprehensive label noise overview is given by (FRÉNAY; VERLEYSEN, 2014).

Individually, label scarcity and label unreliability have been studied extensively, and one can readily find surveys about each subject with pointers to hundreds of original papers (FRÉNAY; VERLEYSEN, 2014; ZHU, 2005b). Despite this, the intersection between them is rather small. This is not unexpected, as both problems pose big challenges, and it would be easy to lose focus when attempting to solve everything in one go. With that said, we find that approaches that do attempt to do this are a very interesting object of study. During the author's Master degree, a systematic review was conducted, revealing that most approaches that addressed this fit within the category of graph-based semi-supervised algorithms. These methods represent each instance as a vertex in a graph, and neighbouring vertices are connected by edges weighted by some similarity metric. This results in a **smoothness criterion** that discourages different predictions in vertices with strong links to each other. We can then regularize our classifier by restricting ourselves to functions that are smooth with respect to the graph, and *label propagation* allows us to spread the known labels to the unlabelled vertices through the graph structure

(ZHU; GHAHRAMANI, 2002).

Previously, we empirically analyzed the performance of different GSSL classifiers subject to label noise (AFONSO; BERTON, 2020) and started developing `LGC_LVOf` (AFONSO, 2020), a novel graph-based semi-supervised filter based on detecting contradictions via a leave-one-out (LOO) approach. However, the original formulation for this filter has a significant disadvantage: namely, that it requires an extra parameter determining the number of instances to be removed. This parameter should ideally be set to the expected number of noisy instances, but in practice this quantity is usually unknown to us. This severely limits the applicability of the original filter. This time around, our focus is on finding better ways to automate our filter, improving its practical value. Our first attempt at improvement was to replace the number of iterations with a more explainable parameter: a threshold, which lets us remove a label only if the internal model assigns its corresponding instance to another class with high enough confidence. This was the method we used in the recently published `LGC_LVOf` paper (AFONSO; BERTON, 2020). This time around, we will attempt to push this further to completely eliminate the necessity for manual parameter tuning.

Our approach is based on gradient descent optimization of the labels themselves. Optimizing the labels directly has been done before: the greedy-gradient method has been applied so as to produce a classifier, the *Graph Transduction through Alternating Minimization* (GTAM) (WANG; JEBARA; CHANG, 2008), which jointly optimizes the final classification and the initial labels. Not much later, this classifier was modified to allow the few nonzero entries in the initial label matrix to deviate from their original value, being renamed to *Label Diagnosis through Self-Tuning* (LDST) (WANG; JIANG; CHANG, 2009). GTAM has been successfully applied in practice (XIU et al., 2018) but has been shown to be unstable under some conditions (AFONSO; BERTON, 2020). Moreover, even when optimized for speed, each step requires $\mathcal{O}(n)$ complexity, and a matrix of size $\mathcal{O}(n^2)$ must be stored in memory. In comparison, the matrix stored by our approach only contains interactions between labelled instances, which are expected to comprise only a small percentage of the dataset.

Our optimization on labels uses a cross-entropy loss, which minimizes the distance to the target distribution. We make an effort to constrain our solution so that labels can only be made weaker, but not change their class outright.

As is often the case with GSSL, all it takes for us to attain good insights is a little bit of linear algebra and matrix calculus. Of course, this is not to say that it is simple work: to understand GSSL, one must understand the language that expresses the relationship between the vertices in a graph, a language complex enough to be able to encompass discrete analogues of heat dissipation and electrical networks. Consequently, the next chapter is devoted to explaining a few key concepts that will prepare us before diving into our filter.

This work is organized into a total of 6 chapters. In this very first chapter, we introduced the problem of learning from data with few and unreliable labels. We also motivated our re-

search by arguing that, although the described scenario may often arise in practice, we seldom find approaches that treat low label availability and reliability in a unified manner. Finally, we further justified the focus of this work: the improvement of our `LGC_LVO` filter, so that it no longer requires a parameter that sets the fixed number of labels to be removed. Our approach, `LGC_LVO_Auto`, takes care of this by removing the dependence on this parameter. Moreover, we also developed an alternative threshold approach to enable the user to have control, while also making a more informed decision. As graph-based methods have been studied extensively for many years, Chapter 2 lays out the necessary concepts for our theoretical analysis. Then, in Chapter 3, we conduct a literature review on semi-supervised methods that explicitly address the possibility of incorrect labels. Chapter 4 can be divided into two parts: firstly, we go over this project's objectives and methodology. What follows is a deep dive into the aforementioned classifier, exploring its behaviour in detail. This includes the proposal of different variants, and also a generalization for this method. We make a point to compare this approach to the one proposed by the author's Master thesis, pointing out differences, similarities, and advantages for either side. With Chapter 5, we put this discussion into practice by implementing those concepts and empirically analyzing results. Lastly, concluding remarks may be found on Chapter 6.

2 Theoretical Framework

The purpose of this chapter is to introduce the necessary concepts for this work. First, we present the well-known field of Machine Learning, with an emphasis on dividing its methods by the criteria of label availability (Section 2.1). From this initial division, we proceed to look at semi-supervised learning, explaining its usefulness and underlying assumptions (Section 2.2). We set our focus to graph-based semi-supervised learning, also known as GSSL (Section 2.4). We also touch upon label noise, including consequences and possible ways to deal with it (Section 2.3). From there, we detail in Section 2.5 previous influential GSSL approaches that are useful to our analysis. This includes the LGC classifier, which effectively acts as the baseline for GTAM and LDST, which are our main object of study. Within the same section, we show that LapRLS can be used as a generalization of LGC, thus justifying the idea of generalizing GTAM by considering the LapRLS framework.

2.1 Supervision in Machine Learning

Machine learning (ML) is the subfield of Computer Science that aims to make a computer learn from data (MITCHELL, 1997). This was defined formally in Chapter 1, where it is stated that learning occurs whenever a previous experience (i.e. observed data) is able to make the program improve its performance in some task. The task we'll be considering is the problem of *classification*, which requires the prediction of a label corresponding to a *class*.

Before the data can be presented to a ML model, it must be represented in some way. Our input is nothing more than a collection of examples, which we denote by

$$\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \tag{2.1}$$

Each object of this collection is called an *instance*. For most practical applications, we may consider an instance to be a vector of d dimensions. As GSSL relies a lot on matrix representations, we will be representing the observed instances as an *input matrix*

$$\mathbf{X} \in \mathbb{R}^{n \times d}. \tag{2.2}$$

As mentioned in Chapter 1, the notion of supervision is a common way to organize the taxonomy of ML classifiers (ALPAYDIN, 2010). Supervision is to be interpreted as an allusion to a teacher showing her students the expected answers for a kind of problem. There are three main paradigms. *Unsupervised learning*, often employed for cluster and prototype identification, does not use even a single label. As such, it cannot be used for classification. *Supervised learning* is at the other end of the spectrum, and it takes for granted the availability

of labels for each and every one of the n instances. The last paradigm we are concerned with (while ignoring some, such as reinforcement learning and self-supervised learning) is *semi-supervised learning*, wherein only some of the labels are known in advance. Once again, it is quite convenient to use a matrix representation, referred to as the *(true) label matrix*

$$\mathbf{Y}_{ij} = \begin{cases} 1 & \text{if the } i\text{-th instance is associated with the } j\text{-th class} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

As the matrix is binary, it is commonly said that $\mathbf{Y} \in \mathbb{B}^{n \times c}$, with c the number of classes. Notice that each row is *one-hot* for labelled instances, that is, exactly one entry is nonzero. For unlabelled instances, the corresponding row is completely null.

All instances are assumed to be drawn independently and identically distributed from some probability distribution $P(\mathbf{x})$. Most ML algorithms work by searching a certain *hypothesis space* $\mathcal{H}$ to find a suitable candidate for $P(y \mid \mathbf{x})$, a function yielding the probability of a label given the input. This function must be consistent with the observed input-output pairs. Failure to do so implies in *underfitting*, and may indicate a lack of complexity in the hypothesis space. Conversely, if we choose a hypothesis space that is too general, there is the risk that our hypothesis is simply remembering each particular case, with no generalization power. This causes *overfitting*. A good ML classifier must avoid both these pitfalls to be useful.

There are a few more distinctions between ML models that are worth pointing out. *Discriminative models* try to model $P(y \mid \mathbf{x})$, known as the class conditional probability. This is considered to be an easier task than modelling the input density $P(\mathbf{x})$ and deriving $P(y \mid \mathbf{x})$ from the application of Bayes' rule (CHAPELLE; SCHÖLKOPF; ZIEN, 2006). The latter is done by *generative models*. The specification of the input distribution is often difficult or intractable. Whenever such a model is able to be built, there is the advantage of being able to sample from the learned joint distribution $P(\mathbf{x}, y)$. Another important distinction is related to the mechanism of generalization, which is a cornerstone of ML. Most approaches are *inductive* in nature, so that we can predict the labels of instances not seen before deployment. However, most GSSL methods are *transductive*, which simply means that we are only interested in the fixed but unknown set of labels corresponding to unlabelled instances (ZHOU et al., 2004). Accordingly, we may represent this with a *classification matrix*:

$$\mathbf{F} \in \mathbb{R}^{n \times c} \tag{2.4}$$

Unlike the initial label matrix, the entries in $\mathbf{F}$ need not be binary. This enables us to interpret each entry $\mathbf{F}_{ij}$ as the confidence in predicting some class to an instance. The classification matrix, once again, has rows corresponding to labelled and unlabelled data, therefore appropriate for working with unreliable labels.

To be consistent with this notation, we will hereafter denote by n the number of instances, d the number of dimensions, c the number of classes.

2.2 Semi-Supervised Learning and its Assumptions

Semi-Supervised learning (SSL) is a ML paradigm designed to address missing labels. To be consistent with our previous discussion, we can assume the set of labels to be relatively small:

$$\mathcal{Y} = \{y_1, \ldots, y_l\}, \quad l \ll n = l + u \tag{2.5}$$

The input instances are divided accordingly:

$$\mathcal{X} = X_{\mathcal{L}} \cup X_{\mathcal{U}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\} \cup \{\mathbf{x}_{(l+1)}, \ldots, \mathbf{x}_{(l+u)}\} \tag{2.6}$$

Using matrix representation, this can be expressed as

$$\mathbf{X} = \left[\mathbf{X}_{\mathcal{L}}{}^{\top}, \mathbf{X}_{\mathcal{U}}{}^{\top}\right]^{\top} \tag{2.7}$$

$$\mathbf{Y} = \left[\mathbf{Y}_{\mathcal{L}}{}^{\top}, \mathbf{Y}_{\mathcal{U}}{}^{\top}\right]^{\top} \tag{2.8}$$

$$\mathbf{F} = \left[\mathbf{F}_{\mathcal{L}}{}^{\top}, \mathbf{F}_{\mathcal{U}}{}^{\top}\right]^{\top} \tag{2.9}$$

The idea of SSL is appealing for many reasons. One of them is the possibility to integrate the toolset developed for unsupervised learning. Namely, we may use unlabelled data to measure the density $P(\mathbf{x})$ within our d-dimensional input space. Once that is achieved, the only thing left is to take advantage of this information. To do this, we have to make use of *assumptions* about the relationship between the input density $P(\mathbf{x})$ and the conditional class distribution $P(y \,|\, \mathbf{x})$. If we are not assuming that our datasets satisfy any kind of assumption, SSL can potentially cause a significant decrease compared to baseline performance (ENGELEN; HOOS, 2020). This is currently an active area of research: *safe semi-supervised learning* is said to be attained when SSL never performs worse than the baseline, for any choice of labels for the unlabelled data. This is indeed possible in some limited circumstances, but it can also be proven to be impossible for others, such as for a specific class of margin-based classifiers (KRIJTHE, 2018). In spite of this pessimistic point of view, there is a silver lining. Namely, that we can seek for weak assumptions that are satisfied by datasets found in practice. Just as many data-producing processes are described adequately by a class of differential equations, one may hope that the datasets themselves exhibit common relationships with their labels.

Even with all of this discussion regarding the usefulness of unlabelled data, it is still not clear what assumptions we should be aiming for. In Figure 1, we illustrate which kind of dataset is suitable for SSL. There are two clear spirals, one corresponding to each class. In a bad dataset for SSL, we can imagine the spiral structure to be a red herring, something that is misguiding. A very common assumption for SSL is the *smoothness assumption*, which is one of the cornerstones for GSSL classifiers:

Assumption 2.2.1 – *Smoothness assumption for semi-supervised learning*
If two instances $\mathbf{x}_1$, $\mathbf{x}_2$ in a high-density region are close, then so should be the corresponding outputs y_1, y_2 (CHAPELLE; SCHÖLKOPF; ZIEN, 2006).

Putting it into context regarding our spiral dataset, we may consider a path between two labels within a spiral. Where input density is high, neighbouring instances are strongly believed to have similar class probabilities. By transitivity, this implies that *geodesics* (i.e. paths) passing exclusively through high-density regions (such as our spirals) are good evidence that the label of those instances is shared. On the other hand, even if we posit that input density is nonzero everywhere, paths between spirals are discouraged by virtue of passing through low-density regions. As a matter of fact, the smoothness assumption turns out to be equivalent as having *decision boundaries* to be located in those regions with low density.

Assumption 2.2.2 – *Low-density separation*
The decision boundary should lie in a low-density region (CHAPELLE; SCHÖLKOPF; ZIEN, 2006).

This duality allows for approaches to be conceived with points of view that are seemingly very different, but ultimately share the same assumption. The GSSL methods put a greater emphasis on using *geodesics* by expressing *connectivity* between instances through the creation of a graph. In contrast, *Transductive Support Vector Machines* (TSVMs) (JOACHIMS, 1999) optimize the *margin* so that it stays in regions of low density. Many successful deep semi-supervised approaches use a similar yet slightly weaker assumption, namely that **small perturbations** in input space should cause little corresponding perturbation on the output space.

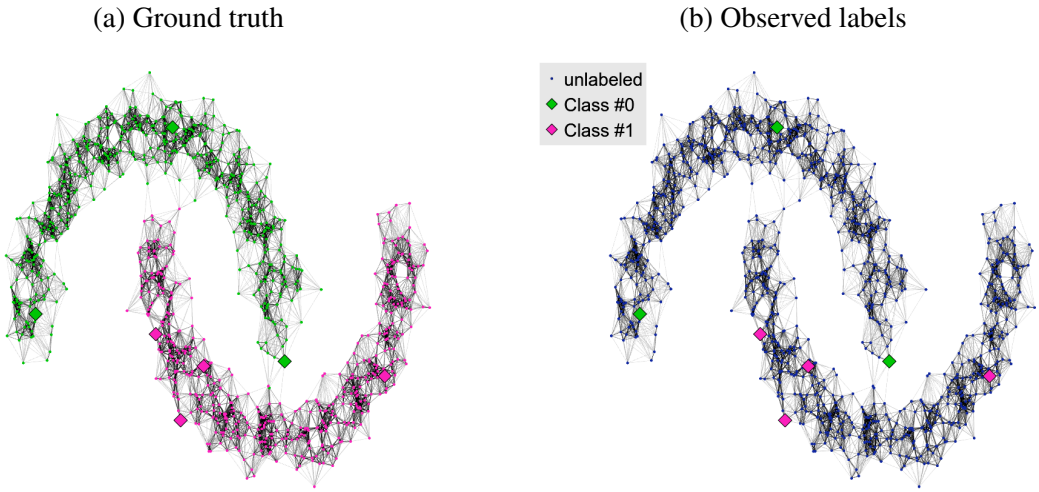


Figure 1: An ideal scenario for semi-supervised learning

There is one last useful assumption. It is very well-known that the approximation of input density is hard if we are working with a very high number of dimensions (BISHOP, 2006). As a result, bounds obtained by the framework of statistical learning get weaker as the

number of dimensions increases. The term *curse of dimensionality* is often employed to refer to such behaviour. However, there is a way to reduce the effect that this has on a classifier. Namely, the manifold assumption.

Assumption 2.2.3 – *Manifold assumption*
The (high-dimensional) data lie (roughly) on a low-dimensional manifold (CHAPELLE; SCHÖLKOPF; ZIEN, 2006).

To justify this, we may consider the problem of digit recognition. Let us consider the portion of the data corresponding to the digit "1". Whenever we increase the width and height of our image, the number of dimensions goes up. This would imply that an absurdly large amount of data must be collected when working with larger images. But let us disregard the resolution of our image for a moment, and try to come up with a better representation. For instance, it is reasonable to believe that most of those digits could be well approximated by specifying rotation, scale, translation, line width, and the length of the short horizontal line at the bottom. As such, we could say that it is located within a low-dimensional *manifold*. The manifold is a structure often associated with studies of Topology that rely on advanced mathematical constructs (LEE, 2010). To us, it is most relevant that this is a space which is locally homeomorphic to $\mathbb{R}^d$. The homeomorphism is a "nice" embedding (as defined by the continuity of its inverse and itself) of this topological space, which is where we expect our data to be, into a Euclidean space of low dimension. The success of GSSL is a sign that many different datasets also satisfy this assumption.

2.3 Modeling Label Noise

As mentioned in the introduction, the other major problem that will be affecting our data quality is label noise. That is, besides already having only a small percentage of labels, they are also not wholly reliable. The problem of label noise has been studied for many decades and originated more publications than one can keep track of. The main reference we have used for this subject is Frénay e Verleysen (2014), a survey which does its best to give a comprehensive overview of label noise on classification problems, referencing around 300 different research papers. Even then, it contains barely any of the SSL approaches of our literature review.

The three major sources of noise are the insufficiency of the description schema, corruption of the input features (*feature noise* or *attribute noise*), and misclassification of training examples (*label noise* or *class noise*) (HICKEY, 1996). Usually, we only consider the first two sources, as measuring the insufficiency of the description schema is usually difficult in practice (ZHU; WU, 2004). Attribute noise is less harmful than label noise (ZHU; WU, 2004) but also more difficult to handle (QUINLAN, 1986).

There aren't many datasets where incorrect labels have been identified. As a result, in

practice noise is artificially injected into the dataset, by corrupting labels $\mathbf{Y}$ by way of some probabilistic process. In the **Noisy Completely at Random Model (NCAR)** model, the probability of an error E is independent of any other random variable, including the true labels $\mathbf{Y}$. This implies that the noise is symmetric: labels of different classes have the same chance to be corrupted, and they are corrupted. For multiclass classification, it is equivalent to flipping a biased coin to determine if a given label is to be corrupted: if the answer is positive, a fair dice is used to determine the new class, original class excluded. If we must adapt the noise process so that a certain class is more likely to be mislabelled, this is called **Noisy At Random (NAR)**. Real-world noise may depend on many things, such as the attributes of an instance. This is referred to as **Noisy Not At Random (NNAR)**.

The taxonomy of label noise approaches is most easily divided into three:

- *Label noise-robust models*: these approaches are usually not designed first and foremost for tackling the problem of label noise. In spite of that, they may end up **robust** to label noise simply because of overfitting avoidance. The decision to restrict the hypothesis space can be efficient at not conforming the model to the noisy labels. The robustness (or lack thereof) may also be related to the loss function. For example, it has been shown that the squared error loss function is tolerant only to uniform noise (MANWANI; SASTRY, 2013).

- *Filter methods*: these have the advantage of addressing label noise before the learning process properly starts. Usually, filters will use some criterion to remove or relabel a suspicious labelled example. A filter that is too sensitive will remove a substantial amount of data, whereas a more conservative filter runs the risk of having little effect on the final classification. Filters are said to be **embedded** when they are built into the classifier model.

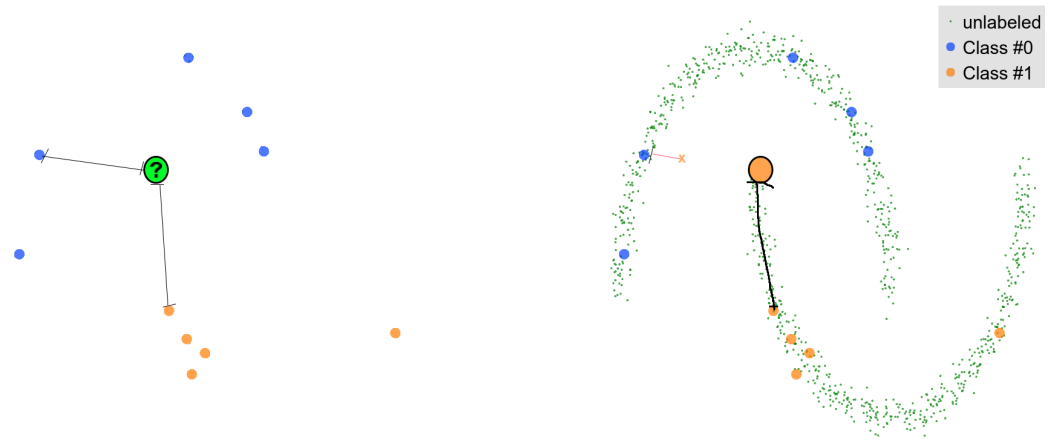
- *Label noise-tolerant methods*: These methods usually take advantage of some prior information available about the label noise at hand. The label noise process is modelled directly. Many of those approaches are probabilistic in nature.

The approach we have proposed in this work can be categorized as a filtering method, which can be naturally embedded into the classifier it is based on. The graph-based semi-supervised methods that surfaced during our literature review usually did not model the distribution of label noise directly. Nonetheless, some of them were explicitly designed to improve robustness to label noise, with very conscious and explicit decisions that aim to minimize the impact of label noise.

2.4 Basics of Graph-based Semi-supervised Learning

Graph-based SSL posits that our data lies on a low-dimensional manifold. It can be argued (CHAPELLE; SCHÖLKOPF; ZIEN, 2006) that using the *geodesic distance* as a metric induced by the manifold is a way to implement the semi-supervised smoothness assumption: the manifold approximates the high-density regions, and two instances are closer with respect to the geodesic distance if they are connected through a high-density region.

The manifold assumption is most useful when the observed embedding in high-dimensional space is curved. Figure 2 shows that, even with only two dimensions, the Euclidean distance can be misleading, whereas the geodesic instance provides a better alternative. If we were to observe similar curves in, say, 100 dimensions, the Euclidean distance gets less discriminative: the curse of dimensionality strikes again, and pairwise distances tend to become more similar (CHAPELLE; SCHÖLKOPF; ZIEN, 2006).



(a) Euclidean distance (b) Geodesic distance

Figure 2: The unlabelled data can be used to reconstruct the manifold, leading to a more expressive metric. Source: (AFONSO, 2020)

Ultimately, the goal is to have a classifier function that is *smooth with respect to the manifold*. Assume that the data lies on a manifold $\mathcal{M}$. The crucial operator when it comes to measuring manifold smoothness is the *Laplace-Beltrami operator*:

$$\Delta f \stackrel{def}{=} div\,(\nabla f) \tag{2.10}$$

It can be shown that $-div$ and ∇ are formally adjoint operators, which simply means that, for any vector field $\mathbf{X}$:

$$\int_{\mathcal{M}} \langle \mathbf{X}, \nabla f \rangle \, dv = \int_{\mathcal{M}} div(\mathbf{X}) f \, dv \tag{2.11}$$

In particular, when $\mathbf{X} = \nabla$, we get

$$S(f) = \int_{\mathcal{M}} \left\|\nabla f\right\|^2 dv = \int_{\mathcal{M}} \Delta(f) f \, dv = \left\langle \Delta(f), f \right\rangle_{L^2(\mathcal{M})} \tag{2.12}$$

This means that, instead of measuring overall manifold smoothness of f by integrating the measure of local smoothness $\left\|\nabla f\right\|^2$ over the whole manifold, we may alternatively take the (infinite-dimensional) inner product between $\Delta(f)$ and f in $L^2(\mathcal{M})$ Hilbert space. A **Hilbert space** $\mathcal{H}$ generalizes the notion of the Euclidean vector space to a possibly infinite number of dimensions, being an inner product space that is complete w.r.t. that inner product. The $L^2(\mathcal{M})$ Hilbert space consists of the functions whose integral over the manifold $\mathcal{M}$ is square-integrable:

$$f : \mathcal{M} \to \mathbb{R} \in L^2(\mathcal{M}) \iff \int_{\mathcal{M}} \left\|f(x)\right\|^2 dx < \infty \tag{2.13}$$

There are a few more notable things about the Laplace-Beltrami operator. First, it is a self-adjoint, positive semidefinite operator. Additionally, it can be shown that the **eigendecomposition** of the Laplace-Beltrami operator provides a basis for all functions in that same Hilbert space. Consequently, every function f has eigendecomposition $f = \sum_i \alpha_i e_i$, where each eigenfunction e_i has a corresponding eigenvalue λ_i. Constant functions have zero eigenvalues, and the remaining are strictly positive. Consequently, the smoothness criterion above is simplified to

$$S(f) = \left\langle \Delta(f), f \right\rangle_{L^2(\mathcal{M})} = \left\langle \sum_i \alpha_i \Delta(e_i), \sum_i \alpha_i e_i \right\rangle_{L^2(\mathcal{M})} = \sum_i \lambda_i \alpha_i^2 \tag{2.14}$$

The key observation here is that any function f is smooth with respect to the manifold, if and only if it is a linear combination whose weights favour mostly eigenfunctions that have small eigenvalues. That is, they should have α_i close (or, preferably, equal) to zero for eigenfunctions with large eigenvalue λ_i. We can therefore force smoothness by restricting our search to functions that are linear combinations of the eigenfunctions with the p smallest eigenvalues, for some fixed p. This is known as **Smooth Eigenbasis Pursuit (SEP)**. Although the discussion above provides insight on how smoothness on manifolds may be defined, we still have to find a way to efficiently compute a solution.

If the manifold is to be considered the underlying intrinsic geometric structure where we observe our input data, then graphs are the tool that provides the best way to approximate said geometric structure. With enough unlabelled data, geodesics are well approximated by paths through the graph. But how should the graph be created? This is very difficult to answer, and we point to (BERTON, 2016) as a thesis dedicated to answering that question. For now, let us focus, again, on Figure 2. If we only look at a small region such as the one encompassing the 3 pink labels of class #1 near each other, we cannot observe the curve of the manifold as much. We can say that, within the neighbourhood of an instance, the Euclidean distance is appropriate. As such, this locality should be exploited. It turns out that we can best express our concepts by defining a measure of *similarity*, instead of distance. In particular, we search for a

affinity matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, such that

$$\mathbf{W}_{ij} = \begin{cases} w(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R} & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are considered neighbours} \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

where w is some function determining the similarity between any two instances $\mathbf{x}_i, \mathbf{x}_j$. When constructing an affinity matrix in practice, instances are not considered neighbours of themselves, i.e. we have $\forall i \in \{1..n\} : \mathbf{W}_{ii} = 0$.

The specification of an affinity matrix is a necessary step for any GSSL classifier, and its sparsity is often crucial for reducing computational costs. There are many ways to choose a neighbourhood. Most frequently, it is constructed by looking at the **K-Nearest Neighbours (KNN)** of a given instance. Let $\mathcal{N}_k(\mathbf{x}_i)$ denote the set of k nearest instances to $\mathbf{x}_i$. A **mutual KNN (mutKNN)** graph has nonzero $\mathbf{W}_{ij}$ whenever $(\mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i) \wedge \mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_j))$. This is more restrictive than the **symmetric KNN (symKNN)**, where the conjunction is turned into a disjunction, and we need only $(\mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i) \vee \mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_j))$. In general, the symmetric version of the KNN is employed, as the properties of some GSSL methods may rely on having a symmetric affinity matrix. We denote the set of pairs $(\mathbf{x}_j, \mathbf{x}_j)$ of connected vertices by E.

Once the neighbourhood structure is defined, one must set a function $w(\mathbf{x}_i, \mathbf{x}_j)$ to distinguish the similarity between $\mathbf{x}_i$ and each of its neighbours $\mathbf{x}_j$. This distinction is not always necessary, and $w(\mathbf{x}_i, \mathbf{x}_j) = 1$ with an appropriate neighbourhood selection can work well in practice, as it relies on neighbours equally. As we have discussed, the Euclidean distance can be locally appropriate. It factors into the **radial basis function (RBF)** kernel, which has the form of a Gaussian and a bandwidth parameter σ that controls how quickly the similarity decreases as instances get farther away:

$$w(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} \tag{2.16}$$

The RBF kernel is a frequently encountered kernel in GSSL literature, partly due to its simplicity. It is a similarity measure that only depends on the distance between instances, and calibration is easier due to having a single parameter σ.

Now that the concepts measuring similarity on a graph have been introduced, they will be the starting point for determining the hypothesis most consistent with the unlabelled data. There are two ways to go about this: we may expect our function to be a result of **label propagation**, an iterative process that spreads the confidence of labels through the graph; or, we take the point of view of optimization, and try to create some cost to be minimized. These often turn out to be equivalent, but we will focus on the latter for now. Given the affinity matrix $\mathbf{W}$, we consider the following measure of local smoothness at a given vertex v:

$$\widetilde{S}_f(v) = \frac{1}{2} \sum_{j:(v,j) \in E} \mathbf{W}_{vj} (f(v) - f(j))^2 \tag{2.17}$$

The analogue measure for overall manifold smoothness (as in Equation (2.12) is

$$\widetilde{S}(f) = \frac{1}{2}\sum_{v\in V}\widetilde{S}_f(v) = \frac{1}{2}\sum_{i\in V}\sum_{j:(i,j)\in E}\mathbf{W}_{ij}(f(i)-f(j))^2 \tag{2.18}$$

Perhaps one of the most pivotal elements of graph-based SSL, the *combinatorial graph Laplacian* is defined as

$$\mathbf{L}_{\mathbb{C}} = \mathbf{D} - \mathbf{W} \tag{2.19}$$

where $\mathbf{D}$, called the *degree matrix*, is a diagonal matrix with entries

$$\mathbf{D}_{ii} = \sum_{j:(i,j)\in E}\mathbf{W}_{ij} \tag{2.20}$$

The graph Laplacian has the same properties as what you'd expect from a discrete analogue of the Laplacian-Beltrami operator. Namely, it satisfies

$$\widetilde{S}(f) = f^\top \mathbf{L}_{\mathbb{C}} f \tag{2.21}$$

and its eigendecomposition provides a basis for the functions on the graph. Fortunately, obtaining the smaller eigenvalues of the graph Laplacian matrix is a well-studied problem, being implemented in most programming libraries for matrix manipulation. Note that, as a result of Equation 2.17, vertices with many similar neighbours are the ones that influence the overall smoothness criterion the most. The way to take care of this is to perform normalization, leading to an alternate local smoothness criterion:

$$\widetilde{S}_{\mathbf{norm}f}(v) = \frac{1}{2}\sum_{j:(v,j)\in E}\left(\frac{\mathbf{W}_{vj}}{\sqrt{\mathbf{D}_{vv}}}f(v) - \frac{\mathbf{W}_{vj}}{\sqrt{\mathbf{D}_{jj}}}f(j)\right)^2 \tag{2.22}$$

This decision in turn leads to a new Laplacian

$$\mathbf{L}_{\mathbb{N}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}_{\mathbb{C}}\mathbf{D}^{-\frac{1}{2}} \tag{2.23}$$

where $\mathbf{I}$ is the identity matrix. Most of the graph-based SSL algorithms will make use of the *unnormalized graph Laplacian* $\mathbf{L}_{\mathbb{C}}$, or the *normalized graph Laplacian* $\mathbf{L}_{\mathbb{N}}$ in some capacity. There are still many other choices for a graph Laplacian operator, such as composing matrices to obtain an *iterated Laplacian*. There is also the *random walk Laplacian*:

$$\mathbf{L}_{\mathbb{R}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W} = \mathbf{D}^{-1/2}(\mathbf{L}_{\mathbb{N}})\mathbf{D}^{1/2} \tag{2.24}$$

It is worth noting that we usually assume that each instance has nonzero similarity with at least one of its neighbours, so that $\mathbf{D}^{-1}$ exists. From now on, we will denote an arbitrary graph Laplacian by $\mathbf{L}$, whenever the choice of graph Laplacian is not relevant to our discussion.

The expression $f^\top \mathbf{L} f$ therefore measures smoothness of f with respect to the graph. A consequence of transduction is that all relevant values of a function that outputs a scalar can be

represented by a column vector. This is remarkable, as it enables us to tackle most problems through linear algebra and matrix calculus. The smoothness criterion appears frequently within GSSL literature, enough for us to include a proof of Equations 2.17 and 2.22. To help with this proof, there are a few lemmas that present some simple facts when working with matrices. Some of them are rather basic, but we believe that having them written down will make our later analysis of GTAM and LDST more clear. For any $i, j \in \mathbb{Z}_+$, we will hereafter denote the vector space of all real matrices of i rows and j columns by $\mathbb{R}^{i\times j}$, for added convenience. We will also use the notation $A[:, k]$ to denote the k-th column of some matrix $\mathbf{A}$, and the $A[k, :]$ the k-th row.

Lemma 2.4.1. *Let $f, g \in \mathbb{R}^{n\times 1}$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$ for some $n \in \mathbb{Z}_+$. Then, it holds that*

$$f^\top \mathbf{A} g = \sum_{i=1}^{n}\sum_{j=1}^{n} f_i \mathbf{A}_{ij} g_j$$

Proof. We use the associative property and matrix composition to get the expansion.

$$\begin{aligned} f^\top \mathbf{A} g &= f^\top (\mathbf{A} g) \\ &= \sum_{i=1}^{n} f_i (\mathbf{A} g)_i \\ &= \sum_{i=1}^{n} f_i \left(\sum_{j=1}^{n} (\mathbf{A}_{ij} g_j) \right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} f_i \mathbf{A}_{ij} g_j \end{aligned}$$

□

Lemma 2.4.2. *Let $\mathbf{A} \in \mathbb{R}^{a\times b}$, $\mathbf{B} \in \mathbb{R}^{b\times c}$, $\mathbf{C} \in \mathbb{R}^{c\times d}$ for given $a, b, c, d \in \mathbb{Z}_+$. Then it holds that $\forall i, j \in \mathbb{Z}_+ : i \le a \wedge j \le d : (\mathbf{ABC})_{ij} = \mathbf{A}_{[i,:]}\mathbf{BC}_{[:,j]}$.*

Proof. Once again, we use the associative property and matrix composition

$$\begin{aligned} (\mathbf{ABC})_{ij} &= (\mathbf{A}(\mathbf{BC}))_{ij} \\ &= \sum_{k=1}^{b} \mathbf{A}_{ik} (\mathbf{BC})_{kj} \\ &= \sum_{k=1}^{b} \mathbf{A}_{ik} \left(\sum_{l=1}^{c} \mathbf{B}_{kl} \mathbf{C}_{l,j} \right) \\ &= \sum_{k=1}^{b}\sum_{l=1}^{c} \mathbf{A}_{ik} \mathbf{B}_{kl} \mathbf{C}_{lj} \end{aligned}$$

If we choose $f = (\mathbf{A}_{[i,:]})^\top$ and $g = \mathbf{C}_{[:,j]}$ as the column vectors, the result follows immediately from lemma 2.4.1. □

Proposition 2.4.3. *Let $n \in \mathbb{Z}_+$. Let $\mathbf{W} \in \mathbb{R}^{n\times n}$ be a symmetric affinity matrix. Define $\widetilde{S}(f) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}(f_i - f_j)^2$. Then, this is the same as $f^\top \mathbf{L}_\mathbb{C} f$.*

Proof. By the definition of $\mathbf{L}_\mathbb{C}$, we have

$$\begin{aligned} f^\top \mathbf{L}_\mathbb{C} f &= f^\top(\mathbf{D} - \mathbf{W})f \\ &= f^\top \mathbf{D} f - f^\top \mathbf{W} f \\ \{\text{Lemma 2.4.1}\} &= \sum_{i=1}^{n}\mathbf{D}_{ii}f_i^2 - \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}f_i f_j \\ \{\text{Equation 2.20}\} &= \left(\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}f_i^2\right) - \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}f_i f_j \end{aligned}$$

The first term can be rewritten as

$$\begin{aligned} &= \left(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}f_i^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ji}f_j^2\right) \\ \{\mathbf{W} \text{ is symmetric}\} &= \left(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}f_i^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}f_j^2\right) \end{aligned}$$

And thus we get

$$\begin{aligned} f^\top \mathbf{L}_\mathbb{C} f &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}(f_i^2 - 2f_i f_j + f_j^2) \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{W}_{ij}(f_i - f_j)^2 \end{aligned}$$

□

Corollary 2.4.3.1. *Let $n \in \mathbb{Z}_+$. Let $\mathbf{W} \in \mathbb{R}^{n\times n}$ be a symmetric affinity matrix. Define $\widetilde{S}_{\mathbf{norm}f}(v) := \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{\mathbf{W}_{ij}}{\sqrt{\mathbf{D}_{ii}}}f(i) - \frac{\mathbf{W}_{ij}}{\sqrt{\mathbf{D}_{jj}}}f(j)\right)^2$. Then, this is the same as $f^\top \mathbf{L}_\mathbb{N} f$.*

Proof. By equation 2.23,we have $\mathbf{L}_\mathbb{N} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}_\mathbb{C}\mathbf{D}^{-\frac{1}{2}}$. This yields $g^\top \mathbf{L}_\mathbb{C} g$ for $g = \mathbf{D}^{-\frac{1}{2}}f$, and the result follows from Proposition 2.4.3. □

The expression $f^\top \mathbf{L} f$ is a cost function related to the smoothness of a column vector f, i.e. a function whose evaluation yields a scalar. Now consider the case where, instead of f, we have a classification matrix $\mathbf{F}$ such that $\mathbf{F}_{ij}$ is proportional to the belief that instance i should be assigned to class j. In this case, we must apply the graph Laplacian to each column individually. The adapted smoothness cost is:

$$\widetilde{S}(\mathbf{F}) = \frac{1}{2}\sum_{k=1}^{c}(\mathbf{F}_{[:,k]})^\top \mathbf{L}(\mathbf{F}_{[:,k]}) = \frac{1}{2}tr(\mathbf{F}^\top \mathbf{L}\mathbf{F}) \tag{2.25}$$

where tr is the trace of the matrix, and the last equality can be obtained by the application of Lemma 2.4.2. This definition works for any graph Laplacian matrix $\mathbf{L}$, including $\mathbf{L}_{\mathbb{C}}$ and $\mathbf{L}_{\mathbb{N}}$.

To ensure smoothness, it is common to use matrix calculus or convex programming to optimize $\widetilde{S}$ directly, often with some restriction. If we get the eigendecomposition of our solution, some of the less smooth eigenfunctions are still used, even if their weight is expected to be much lower. The only way to actually restrict our basis is with the previosly mentioned smooth eigenbasis pursuit, which serves as a complementary regularization.

Calculating the smoothness of a function with respect to the graph is an incredibly useful tool. With it, we restrict the hypothesis space of candidate functions, or discourage functions that have changes in high-density regions. However, it is not enough on its own. After all, any constant function will be perfectly smooth, and yet not what one is looking for. To address this, any GSSL approach must also ensure that the function is consistent with the observed labels, by way of some *label fitting criterion*. There are many ways to find a function that it is smooth but also fits the labels accordingly. Next, we will see two classic GSSL approaches and how they tackle the issue of balancing these two objectives.

2.5 Some GSSL classifiers and a generalization

In this section, we will introduce three seemingly very distinct GSSL classifiers. These are among the most cited approaches to be found in GSSL literature. Notably, it can be shown that there is a framework which generalizes all of these classifiers (SOUSA, 2017). This is quite remarkable, given that our objective is to generalize the bivariate formulation provided by GTAM and LDST.

2.5.1 GFHF: Gaussian Fields and Harmonic Functions

The *Gaussian Fields and Harmonic Functions* (GFHF) (ZHU; GHAHRAMANI, 2002) classifier assumes that label information is perfect, and optimizes for graph smoothness around this restriction. Let $\widehat{\mathbf{Y}}^{(0)} = \mathbf{Y} = \left[\mathbf{Y}_{\mathcal{L}}{}^{\top}, \mathbf{Y}_{\mathcal{U}}{}^{\top}\right]^{\top}$ be the initial label matrix. Label propagation occurs via the iteration of the following:

$$\widehat{\mathbf{Y}}^{(t+1)} = \widetilde{\mathbf{W}}\,\widehat{\mathbf{Y}}^{(t)}; \quad \widehat{\mathbf{Y}_{\mathcal{L}}}^{(t+1)} = \mathbf{Y}_{\mathcal{L}} \tag{2.26}$$

Here, $\widetilde{W}$ is the row-normalized weight matrix, defined as $\widetilde{\mathbf{W}} = \mathbf{D}^{-1}\mathbf{W}$. The purpose of row normalization is to get transition probabilities. Due to clamping, labelled instances act as sink states. Computing the limit of the configuration of this random walk is thus the same as figuring out which class is likely reached first. This random walk has the advantage of not requiring some parameter r which controls the number of steps taken, such as in Szummer e Jaakkola (2002). Moreover, it can be shown that this algorithm imposes $\mathbf{F}_{\mathcal{L}} = \mathbf{Y}_{\mathcal{L}}$ and then minimizes

the smoothness criterion $\mathbf{F}^\top \mathbf{L}_\mathbb{C} \mathbf{F}$. One can obtain the equilibrium in closed form, namely

$$\mathbf{F}_\mathcal{U} = (\mathbf{I} - \widetilde{\mathbf{W}}_{\mathcal{U}\mathcal{U}})^{-1} \widetilde{\mathbf{W}}_{\mathcal{U}\mathcal{L}} \mathbf{Y}_\mathcal{L} = ((\mathbf{L}_\mathbb{C})_{\mathcal{U}\mathcal{U}})^{-1} (\mathbf{L}_\mathbb{C})_{\mathcal{U}\mathcal{L}} \mathbf{Y}_\mathcal{L} \tag{2.27}$$

The naming of this approach owes to the *harmonic property*, satisfied by the GFHF solution. Namely, that its value at each unlabelled instance is a weighted average of its neighbours at unlabelled points. The weights are exactly the entries in $\widetilde{\mathbf{W}}$. This is equivalent to:

$$(\mathbf{L}_\mathbb{C} \mathbf{F})_\mathcal{U} = 0 \tag{2.28}$$

2.5.2 LGC: Local and Global Consistency

The *Local and Global Consistency* (LGC) (ZHOU et al., 2004) algorithm is one of the most widely known graph-based semi-supervised algorithms. The cost being minimized is as follows:

$$\mathcal{Q}(\mathbf{F}) = \frac{1}{2} \left(tr(\mathbf{F}^\top \mathbf{L}_\mathbb{N} \mathbf{F}) + \mu \left\| \mathbf{F} - \mathbf{Y} \right\|^2 \right) \tag{2.29}$$

The first difference between GFHF and LGC is that the latter normalizes its graph Laplacian (Equation 2.23). LGC also mitigates the issue of label reliability by introducing the parameter $\mu \in (0, \infty)$. This parameter controls the trade-off between a function that fits the labels exactly, and one which attains high graph smoothness.

LGC employs a cost function which is convex. The analytic solution is found by first taking the partial derivative of $\mathcal{Q}$ with respect to $\mathbf{F}$:

$$\frac{\partial \mathcal{Q}}{\partial \mathbf{F}} = \frac{1}{2} \frac{\partial tr(\mathbf{F}^\top \mathbf{L}_\mathbb{N} \mathbf{F})}{\partial \mathbf{F}} + \frac{1}{2} \mu \frac{\partial \left\| \mathbf{F} - \mathbf{Y} \right\|^2}{\partial \mathbf{F}} \tag{2.30}$$

$$= \mathbf{L}_\mathbb{N} \mathbf{F} + \mu (\mathbf{F} - \mathbf{Y}) \tag{2.31}$$

$$= (\mathbf{I} - \mathbf{S}) \mathbf{F} + \mu \mathbf{F} - \mu \mathbf{Y} \tag{2.32}$$

$$= ((1 + \mu) \mathbf{I} - \mathbf{S}) \mathbf{F} - \mu \mathbf{Y} \tag{2.33}$$

where $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{L}_\mathbb{N}$. By dividing the above by $(1 + \mu)$, we observe that this derivative is zero exactly when

$$(I - \alpha \mathbf{S}) \mathbf{F} = \beta \mathbf{Y} \tag{2.34}$$

with

$$\alpha = \frac{1}{1 + \mu} \in (0, 1) \tag{2.35}$$

and

$$\beta = 1 - \alpha \tag{2.36}$$

The matrix $(\mathbf{I} - \alpha \mathbf{S})$ is positive-definite. To verify this, let f be any column vector of size n that is not null. Then,

$$f^\top (\mathbf{I} - \alpha \mathbf{S}) f = f^\top \frac{1}{1 + \mu} (\mathbf{L}_\mathbb{N} + \mu \mathbf{I}) f \tag{2.37}$$

$$= \frac{1}{1 + \mu} \left(f^\top \mathbf{L}_\mathbb{N} f + \mu f^\top f \right) \tag{2.38}$$

In this expression, $\frac{1}{1+\mu}$ and μ are positive constants. The first term is non-negative due to the positive semi-definiteness of $\mathbf{L}_{\mathbb{N}}$, and the second term has to be positive for any non-null f. Any positive-definite matrix is invertible, so the optimal $\mathbf{F}$ can be obtained as

$$\mathbf{F} = \beta(I - \alpha\mathbf{S})^{-1}\mathbf{Y} \tag{2.39}$$

We hereafter refer to $(I - \alpha S)^{-1}$ as the **propagation matrix** $\mathbf{P}$. Each entry $\mathbf{P}_{ij}$ represents the amount of label information from X_j that X_i inherits. It can be shown that the inverse is a result of a diffusion process, which is calculated via iteration:

$$F(0) = \mathbf{Y} \tag{2.40}$$

$$F(t{+}1) = \alpha\mathbf{S}F(t) + (1{-}\alpha)\mathbf{Y} \tag{2.41}$$

Moreover, it can be shown that the closed expression for F at any iteration is

$$F(t) = (\alpha\mathbf{S})^{t-1}\mathbf{Y} + (1-\alpha)\sum_{i=0}^{t-1}(\alpha\mathbf{S})^i\mathbf{Y} \tag{2.42}$$

S is similar to $D^{-1}W$, whose eigenvalues are always in the range $[-1, 1]$ (ZHOU et al., 2004). We assume that $\alpha \in (0, 1)$, which ensures the first term vanishes as t grows larger, whereas the second term converges to $\mathbf{P}Y$. Consequently, $\mathbf{P}$ can be characterized as

$$\mathbf{P} = (1-\alpha)\lim_{t\to\infty}\sum_{i=0}^{t}(\alpha\mathbf{S})^i \tag{2.43}$$

$$= (1-\alpha)\lim_{t\to\infty}\sum_{i=0}^{t}\alpha^i\left(\mathbf{D}^{\frac{1}{2}}(\mathbf{D}^{-1}\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\right)^i \tag{2.44}$$

$$= (1-\alpha)\lim_{t\to\infty}\sum_{i=0}^{t}\alpha^i\mathbf{D}^{\frac{1}{2}}(\mathbf{D}^{-1}\mathbf{W})^i\mathbf{D}^{-\frac{1}{2}} \tag{2.45}$$

The **transition probability matrix** $\widetilde{W} = \mathbf{D^{-1}W}$ makes it so we can interpret the process as a random walk. Let us imagine a particle walking through the graph according to the transition matrix. Assume it began at a labelled vertex v_a, and at step i it reaches a labelled vertex v_b, initially labelled with class c_b. When this happens, v_a receives a confidence boost to class c_b. This boost is proportional to α^i. This gives us a good intuition as to the role of α. More precisely, the contribution of vertices found later in the random walk decays exponentially according to α^i.

2.5.3 LapRLS: Laplacian Regularized Least Squares

Kernel methods are a class of machine learning algorithms which make use of *kernel functions*. One motivation for those methods is that the data in practice may not be linearly separable, at least in the given input space. Thus, it would be desirable to have a *feature map* ϕ

that projects our data in $\mathcal{X}$ to a higher-dimensional space where a separating hyperplane can be found. Unfortunately, specifying ϕ directly is not obvious. However, the second motivation for these methods is that, to perform empirical risk minimization, it is enough to know the *inner product* between the image of the instances under ϕ.

A central concept to kernel methods is the notion of **Reproducing Kernel Hilbert Spaces (RKHS)**. For simplicity, we will hereafter assume that each element of the Hilbert space $\mathcal{H}$ we are working with is a function $f : \mathcal{X} \to \mathbb{R}$ from our input space to a real number. If we are dealing with a binary classification problem, we will consider this to be a hypothesis space where we must find a suitable classification function f. A RKHS requires that, for every $x \in \mathcal{X}$, the linear evaluation functional to be bounded by some $m \in \mathbb{R}$. That is, if we define $E_x : \mathcal{H} \to \mathbb{R}$ as $E_x(f) = f(x)$, then

$$\exists m \in \mathbb{R} : \forall x \in \mathcal{X}, f \in \mathcal{H} : E_x(f) \leq m\big\|f\big\|_{\mathcal{H}} \tag{2.46}$$

It is not immediately obvious as to why requiring this operator to be bounded is what defines the RKHS. The reason is that, by the Riesz Representation Theorem, this is sufficient for the **reproducing property** to hold. This property asserts that, for each $x \in \mathcal{X}$, there exists $\phi : \mathcal{X} \to \mathcal{H}$ such that:

$$\forall x \in \mathcal{X}, f \in \mathcal{H} : E_x(f) = f(x) = \big\langle \phi(x), f \big\rangle_{\mathcal{H}} \tag{2.47}$$

In other words, if we interpret ϕ as feature map, any function within this Hilbert space can be evaluated at x by taking the inner product with its representative element $\phi(x)$.

If we did have the knowledge of some $\phi : \mathcal{X} \to \mathcal{H}$ for some Hilbert space $\mathcal{H}$ over $\mathbb{R}$, then a function $ker : \mathcal{X} \times \mathcal{X} \to \mathcal{H}$ satisfying

$$\forall x, y \in \mathcal{X} : ker(x, y) = \big\langle \phi(x), \phi(y) \big\rangle_{\mathcal{H}} \tag{2.48}$$

is called a **kernel**. If we have n observed instances in $\mathcal{X}$, we may index all kernel values by a **kernel matrix** $\mathbf{K} \in \mathbb{R}^{n \times n}$. Can we just ignore ϕ and implictly define a RKHS by hand-picking the kernel values directly? Not always. As it turns out, we need the kernel to be symmetric and positive semi-definite. Equivalently, our matrix $\mathbf{K}$ must satisfy

$$(\mathbf{K} = \mathbf{K}^\top) \ \wedge \ \big(\forall a \in \mathbb{R}^{n \times 1} : a^\top \mathbf{K} a \geq 0\big) \tag{2.49}$$

If these two restrictions are satisfied, we say that it is a **positive definite kernel**. It can be shown that there is indeed a one-to-one correspondence between a positive definite kernel and some (implicitly defined) RKHS, whose feature map ϕ satisfies

$$\forall x \in \mathcal{X} : (\phi(x) = ker(\cdot, x) \in \mathcal{H}) \tag{2.50}$$

and, crucially, having the reproducing property leads to one of the most important theorems in ML:

Theorem 2.5.1 (Representer Theorem)**.** *Let ker be a symmetric positive-definite kernel function, which corresponds to some RKHS $\mathcal{H}$ with norm denoted by $\|\cdot\|_K^2$. Let V be any loss function. Then, the problem of empirical risk minimization*

$$f^* = argmin_{f\in\mathcal{H}} \left(\frac{1}{l}\sum_{i=1}^{l} V(x_i, y_i, f) + \lambda \|f\|_K^2 \right)$$

Has a solution of the form

$$f^*(x) = \sum_{i=1}^{l} a_i ker(x, x_i)$$

for some $a = (a_1, \ldots, a_l)$.

This formulation is the pillar of many supervised ML staples, such as the least squares classifier and support vector machines. If we define a suitable kernel matrix, the Representer Theorem assures that we can represent the solution as a linear combination of the similarities $ker(x_i, \cdot)$ to each labelled instance x_i. This greatly simplifies the optimization procedure for nontrivial classes of functions, as we only need to optimize the coefficients $a_1, \ldots, a_l$. In order to turn any such classifier into the paradigm of SSL (BELKIN; NIYOGI; SINDHWANI, 2006), the manifold assumption will be incorporated by adding a regularization term similar to Equation 2.12. The data in reality may not lie on a manifold, so the authors use the concept of a measure $d\mathcal{P}_x$ on the underlying probability distribution. We end up with a similar result, adapted to SSL:

Theorem 2.5.2 (Representer Theorem for SSL)**.** *Let ker be a symmetric positive-definite kernel function, which corresponds to some RKHS $\mathcal{H}$ with norm denoted by $\|\cdot\|_K^2$. Let V be any loss function. Moreover, let $\|\cdot\|_I^2$ be an intrinsic regularization term satisfying*

$$\|\cdot\|_I^2 = \int_{\mathcal{M}} A(f) f \, d_{\mathcal{P}_x} = \langle f, Af \rangle_{L^2(\mathcal{P}_x)}$$

where $\mathcal{M}$ is the support of $\mathcal{P}_x$ (in most cases, this corresponds to the manifold the data lies in), and $A : \mathcal{H} \to L^2(\mathcal{P}_x)$ is a bounded operator. Then, the problem of semi-supervised empirical risk minimization

$$f^* = argmin_{f\in\mathcal{H}} \left(\frac{1}{l}\sum_{i=1}^{l} V(x_i, y_i, f) + \lambda_A \|f\|_K^2 + \lambda_I \|f\|_I^2 \right)$$

has a solution of the form

$$f^*(x) = \sum_{i=1}^{l} a_i k(x, x_i) + \int_{\mathcal{M}} \widetilde{a}(y) K(x, y) d_{\mathcal{P}_x(y)}$$

for some $a = (a_1, \ldots, a_l)$ and $\widetilde{a} : \mathcal{M} \to \mathbb{R}$.

It is important to note that the Laplace-Beltrami operator satisfies the necessary condition for A in Theorem 2.5.2. Moreover, as we usually have to estimate the density $\mathcal{P}_x$ from

unlabelled examples, some graph Laplacian $\mathbf{L}$ will replace the Laplace-Beltrami operator, yielding:

$$f^* = \text{argmin}_{f \in \mathcal{H}} \left(\frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \lambda_A \|f\|_K^2 + \frac{\lambda_I}{n^2} f^\top \mathbf{L} f \right) \tag{2.51}$$

After discretization, the solution remains with the general form of the one from Theorem 2.5.2, which means that for a transductive approach the solution will satisfy

$$\mathbf{F} = \mathbf{K}\mathbf{a} \quad \mathbf{a} \in \mathbb{R}^{n \times 1} \tag{2.52}$$

The Representer Theorem is a very strong result that gave rise to a whole class of kernelized methods, which take advantage of an immensely powerful and robust framework based on the theory of RKHS. The **Laplacian Regularized Least Squares (LapRLS)** is one such algorithm, that follows from choosing the loss function to be the usual squared loss. The updated cost function is therefore:

$$\mathcal{Q}(\mathbf{a}) = \left(\frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \lambda_A \mathbf{a}^\top \mathbf{K}\mathbf{a} + \frac{\lambda_I}{n^2} \mathbf{a}^\top \mathbf{K}\mathbf{L}\mathbf{K}\mathbf{a} \right) \tag{2.53}$$

As we'll see next, LapRLS can be slightly modified to allow for the generalization of LGC and GFHF.

2.5.4 LGC and GFHF are generalized by the LapRLS Framework

This section is a summary of some results found in (SOUSA, 2017), which provides us a formulation that enables a generalization of Local and Global Consistency, as well as Gaussian Fields and Harmonic Functions. Both the LGC and GFHF are mainstay methods within the application of GSSL. It is easy to note that both these functions use a *smoothness criterion* and a *label fitting* criterion to find a suitable transductive solution. In spite of that, there are some key differences: LGC opts for the normalized graph Laplacian $\mathbf{L}_\mathbb{N}$ and a parameter μ controlling the trade-off between criterions; GFHF uses $\mathbf{L}_\mathbb{C}$ and locks the transductive output on labelled instances, so that they remain untouched. Regardless, it can be shown that, if the frameworks of the original LGC and GFHF approaches are slightly tweaked, then LGC generalizes GFHF. In order to attain greater generalization, the first thing to is to disregard the specific choice of graph Laplacian $\mathbf{L}$. Secondly, the regularization term, previously denoted by μ (Equation 2.29), is now replaced with a diagonal matrix $\mathbf{\Sigma}$ which we will name here as the **regularization matrix**:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma_l} & \mathbf{0_{l \times u}} \\ \mathbf{0_{u \times l}} & \mathbf{\Sigma_u} \end{bmatrix} \tag{2.54}$$

The slightly tweaked framework for LGC is then

$$\mathcal{Q}(\mathbf{F}) = tr\left(\mathbf{F}^\top \mathbf{L}\mathbf{F} + (\mathbf{F} - \mathbf{Y})^\top \mathbf{\Sigma} (\mathbf{F} - \mathbf{Y}) \right) \tag{2.55}$$

The original LGC formulation of (ZHOU et al., 2004) is the particular case where $\mathbf{\Sigma} = \mu\mathbf{I_{n\times n}}$ and $\mathbf{L} = \mathbf{L}_{\mathbb{C}}$. The solution, in closed form, is given as

$$\mathbf{F} = (\mathbf{L} + \mathbf{\Sigma})^{-1}\mathbf{\Sigma Y} \tag{2.56}$$

The expression $\mathbf{\Theta} = (\mathbf{L} + \mathbf{\Sigma})^{-1}$ is an alternate definition for the propagation matrix. Note that this matrix is not exactly equal to the definition used by Equation 2.43:

$$(\mathbf{I} + \Sigma)\mathbf{\Theta} = \mathbf{P} \tag{2.57}$$

Therefore, the choice of a new symbol to represent this alternate propagation matrix is justified.

The GFHF classifier may be slightly generalized to include an arbitrary graph Laplacian $\mathbf{L}$, being formulated as the following problem.

$$\text{argmin}_{\mathbf{F}\in\mathbb{R}^{n\times c}} tr(\mathbf{F}^\top\mathbf{L}\mathbf{F}) \quad \text{s.t. } \mathbf{F}_\mathcal{L} = \mathbf{Y}_\mathcal{L} \tag{2.58}$$

This yields a solution much like 2.27, but replacing $\mathbf{L}_{\mathbb{C}}$ with any graph Laplacian $\mathbf{L}$

$$\mathbf{F}_\mathcal{L} = \mathbf{Y}_\mathcal{L} \tag{2.59}$$

$$\mathbf{F}_\mathcal{U} = (\mathbf{L}_{\mathcal{U}\mathcal{U}})^{-1}\mathbf{L}_{\mathcal{U}\mathcal{L}}\mathbf{Y}_\mathcal{L} \tag{2.60}$$

Notably, it can be shown that the following proposition holds:

Proposition 2.5.1. *(GFHF as a special case of LGC) Let LGC be defined as in Equation 2.55. Then, if* $\mathbf{\Sigma_u} \to \mathbf{0}_{\mathbf{l}\times\mathbf{l}}$ *and* $\mathbf{\Sigma_u} \to \mathbf{0}_{\mathbf{l}\times\mathbf{l}}$*, then the function* $\mathbf{F}$ *that minimizes that cost function also approximates the solution of the GFHF problem (Equation 2.58).*

Proof. Refer to (SOUSA, 2017).

Lastly, the LapRLS classifier also receives some minor modifications. The new formulation once again introduces a regularization matrix $\mathbf{\Sigma}$, but also generalizes to a multi-class output:

$$\text{argmin}_{\mathbf{a}\in\mathbb{R}^{n\times c}} tr\left((\mathbf{Ka} - \mathbf{Y})^\top\mathbf{\Sigma}(\mathbf{Ka} - \mathbf{Y}) \;+\; \lambda_A\mathbf{a}^\top\mathbf{Ka} \;+\; \lambda_I\mathbf{a}^\top\mathbf{KLKa}\right) \tag{2.61}$$

as a result, the transductive solution is

$$\mathbf{F} = \mathbf{Ka} \tag{2.62}$$

such that

$$\mathbf{a} = (\mathbf{\Sigma K} + \lambda_A\mathbf{I_{n\times n}} + \lambda_I\mathbf{LK})^{-1}\mathbf{\Sigma Y} \tag{2.63}$$

Notice how, if we plug $\lambda_A = 0, \lambda_I = 1$, this solution is reduced to Equation 2.56. As such, this framework for LapRLS generalizes the ones for LGC and GFHF. In particular, the two previous methods are special cases where the kernel matrix $\mathbf{K}$ is entirely ignored. As noted in Belkin, Niyogi e Sindhwani (2006), the term λ_A is desirable for situations where the manifold assumption does not hold. Therefore, we find this to be an indicator that LapRLS should perform better than the other two methods when the data violates this assumption.

2.5.5 Convergence Restrictions in the Generalized LGC

In spite of being able to generalize to multiple scenarios, there are a few restrictions to the framework introduced in Section 2.5.4. Namely, there are some requirements necessary for convergence.

Let us consider the spectrum of graph Laplacians. Recall the random walk Laplacian $\mathbf{L}_{\mathbb{R}}$ (Equation 2.24). It is similar to the normalized graph Laplacian $\mathbf{L}_{\mathbb{N}}$, sharing its eigenvalues with it. In addition, it is defined as $\mathbf{I} - \widetilde{\mathbf{W}}$, where $\widetilde{\mathbf{W}} = \mathbf{D}^{-1}\mathbf{W}$ is a row-stochastic matrix. As such, every eigenfunction of $\mathbf{L}_{\mathbb{R}}$ will be an eigenfunction of $\widetilde{\mathbf{W}}$, with the new eigenvalue equal to one minus the old eigenvalue. Fortunately, it is not hard to show that every row-stochastic matrix has a spectrum in the range $[-1, 1]$.

Lemma 2.5.2. *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be any matrix whose entries are non-negative. Assume this matrix is row-stochastic, i.e. $\mathbf{A}\mathbf{1}_n = \mathbf{1}_n$. Then its eigenvalues are in the range $[-1, 1]$.*

Proof. Assume the contrary. Then, there is $\mathbf{v}$ such that $\mathbf{A}\mathbf{v} = c\mathbf{v}$ with $\|c\| > 1$. Assume, without loss of generality, that $\mathbf{v}_i$ is the value with largest absolute value within the vector. Because $\mathbf{A}$ is row-stochastic, by definition $(\mathbf{A}\mathbf{v})_i$ holds some convex combination, whose norm is given by

$$\|\sum_k^n a_{ij}\mathbf{v}_k\| \leq \sum_j^n \|a_{ij}\mathbf{v}_j\| \leq \sum_j^n a_{ij}\|\mathbf{v}_i\| = (\sum_j^n a_{ij})\|\mathbf{v}_i\| = 1\,\mathbf{v}_i < c\|\mathbf{v}_i\|$$

Thus, we arrive at a contradiction. We conclude that all eigenvalues lie in the range [-1,1] □

Corollary 2.5.2.1. *The eigenvalues of $\widetilde{\mathbf{W}} = \mathbf{D}^{-1}\mathbf{W}$ lie in [-1,1]. The eigenvalues of $\mathbf{L}_{\mathbb{N}}$ and $\mathbf{L}_{\mathbb{R}}$ lie in [0,2].*

For symmetric matrices (we assume $\mathbf{W}$ is symmetric, so this holds for all the graph Laplacians) the **Rayleigh quotient** bounds the spectrum:

$$R(\mathbf{L}, f) = \frac{f^\top \mathbf{L} f}{f^\top f} \tag{2.64}$$

The unnormalized spectrum is much harder to deal with. Assume that g is a unit-norm eigenvector of $\mathbf{L}_{\mathbb{N}}$, and $g = \mathbf{D}^{\frac{1}{2}} f$. Then

$$\begin{aligned} R(\mathbf{L}_{\mathbb{C}}, f) &= \frac{f^\top \mathbf{L}_{\mathbb{C}} f}{f^\top f} \\ &= \frac{f^\top \mathbf{D}^{\frac{1}{2}} \mathbf{L}_{\mathbb{N}} \mathbf{D}^{\frac{1}{2}} f}{f^\top f} \\ &= \frac{g^\top \mathbf{L}_{\mathbb{N}} g}{(\mathbf{D}^{-\frac{1}{2}} g)^\top (\mathbf{D}^{-\frac{1}{2}} g)} \\ &\leq \frac{2}{\sum_{k=0}^n g_{ii}/D_{ii}} \end{aligned}$$

This indicates that the bounds on the unnormalized Laplacian may depend on the degree of each instance. For this reason, we will avoid using $\mathbf{L}_{\mathbb{C}}$.

Lastly, it is important to emphasize that many of the solutions we have seen so far are a kind of **Neumann series** for some matrix $\mathbf{T}$:

$$(\mathbf{I} - \mathbf{T})^{-1} = \sum_{i=0}^{\infty} \mathbf{T}^i \tag{2.65}$$

For the convergence of a Neumann series, it is sufficient condition to have a **spectral radius** $\rho(\mathbf{T})$ with a value less than 1. The spectral radius is simply the largest absolute value of an eigenvalue for a given matrix. For the generalized GFHF specifically, it suffices that $\rho(\mathbf{I} - \mathbf{L}_{uu}) < 1$. For the case $\mathbf{L} = \mathbf{L}_{\mathbb{C}}$, all eigenvalues lie inside the interval $[0, 2]$.

For the **generalized GFHF**, it is shown by Sousa (2017) that it is sufficient to have $\rho(\mathbf{I}_{\mathcal{U}} - \mathbf{L}_{\mathcal{U}\mathcal{U}}) > 1$. More worrisome is the fact that, for the **generalized LGC**, the eigenvalues of $\mathbf{\Sigma}^{-1}\mathbf{L}$ must be inside [0,1]. As a matter of fact, the propagation matrix $\mathbf{\Theta}$ can be rewritten as a Neumann series

$$\mathbf{\Theta} = (\mathbf{L} + \mathbf{\Sigma})^{-1} = \mathbf{\Sigma}^{-1}(\mathbf{I} + \mathbf{\Sigma}^{-1}\mathbf{L})^{-1} \tag{2.66}$$

which will not converge if $\rho(-\mathbf{\Sigma}^{-1}\mathbf{L}) > 1$. At first glance, this is somewhat strange, as the original LGC algorithm allowed for any positive value of μ. Here, on the other hand, the corresponding $\mathbf{\Sigma} = \mu\mathbf{I}$ makes the series diverge for small values of μ. To be sure of this, simply note that $\frac{1}{\mu}$ serves as a constant that multiplies each eigenvalue of $\mathbf{L}$. In practice, small values of μ are preferred, because they allow the vertices that are reached later to still yields some reward. As such, it seems that the "generalized LGC" actually restricts some of the most important subsets of the solution space. In light of this, we will try to provide a better generalization.

We start from the general cost 2.55. Let $\boldsymbol{\alpha} \in \mathbb{R}^{n \times n}$ be a diagonal matrix whose values are the ones in $\mathbf{\Sigma}$ after applying the function $f(\mu) = \frac{1}{1+\mu}$. That is, $\boldsymbol{\alpha} = (\mathbf{I} + \mathbf{\Sigma})^{-1}$. It is also true that $\mathbf{\Sigma}(\mathbf{I} - \boldsymbol{\alpha}\mathbf{S})^{-1} = (\mathbf{I} - \boldsymbol{\alpha})$. Taking the derivative with respect to the cost and equating it to zero yields

$$\begin{aligned}
&\mathbf{L}\mathbf{F} + \mathbf{\Sigma}(\mathbf{F} - \mathbf{Y}) = 0 \\
\iff &\mathbf{L}\mathbf{F} + \mathbf{\Sigma}\mathbf{F} = \mathbf{\Sigma}\mathbf{Y} \\
\iff &(\mathbf{I} - \mathbf{S})\mathbf{F} + \mathbf{\Sigma}\mathbf{F} = \mathbf{\Sigma}\mathbf{Y} \\
\iff &(\mathbf{I} - \mathbf{S} + \mathbf{\Sigma})\mathbf{F} = (\mathbf{L} + \mathbf{\Sigma})\mathbf{F} = \mathbf{\Sigma}\mathbf{Y} \\
\iff &(\mathbf{I} + \mathbf{\Sigma})^{-1}(\mathbf{I} - \mathbf{S} + \mathbf{\Sigma})\mathbf{F} = (\mathbf{I} + \mathbf{\Sigma})^{-1}\mathbf{\Sigma}\mathbf{Y} \\
\iff &(\boldsymbol{\alpha} - \boldsymbol{\alpha}\mathbf{S} + (\mathbf{I} + \mathbf{\Sigma})^{-1}\mathbf{\Sigma})\mathbf{F} = (\mathbf{I} + \mathbf{\Sigma})^{-1}\mathbf{\Sigma}\mathbf{Y} \\
\iff &(\boldsymbol{\alpha} - \boldsymbol{\alpha}\mathbf{S} + \mathbf{I} - \boldsymbol{\alpha})\mathbf{F} = (\mathbf{I} - \boldsymbol{\alpha})\mathbf{Y} \\
\iff &(\mathbf{I} - \boldsymbol{\alpha}\mathbf{S})\mathbf{F} = (\mathbf{I} - \boldsymbol{\alpha})\mathbf{Y}
\end{aligned}$$

All of these logical connectives are biconditional, so the solution $\mathbf{F} = (\mathbf{I} - \alpha\mathbf{S})^{-1}(\mathbf{I} - \boldsymbol{\alpha})\mathbf{Y}$ minimizes the generalized LGC cost. The matrix $(\mathbf{I} - \boldsymbol{\alpha}\mathbf{S})$ is referred to as $\boldsymbol{\Upsilon}$ in Sousa (2017).

Proposition 2.5.3 (Convergence of the modified generalized LGC)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a diagonal matrix such that $0 \leq \max\{\boldsymbol{A}_{ii} \mid 1 \leq i \leq n\} < 1$. Let $\mathbf{S} \in \mathbb{R}^{n\times n}$ be a matrix with only nonnegative entries, such that $\rho(\mathbf{S}) < 1$. Then $(\mathbf{I} - \boldsymbol{A}\mathbf{S})^{-1}$ is invertible.*

Proof. As this is a Neumann series, it is sufficient to have

$$\sum_{t=0}^{\infty}(\boldsymbol{A}\mathbf{S})^t \tag{2.67}$$

converge. The result follows inductively from observing that, if $\mathbf{B} \in \mathbb{R}^{n\times n}$ has exclusively nonnegative entries, then so will $\boldsymbol{A}\mathbf{A}$, and every entry in $\boldsymbol{A}\mathbf{B}$ is at most $(\max\{\boldsymbol{A}_{ii} \mid 1 \leq i \leq n\})\mathbf{B}$, therefore

$$\mathbf{0}^{n\times n} \leq \sum_{t=0}^{\infty}(\boldsymbol{A}\mathbf{S})^t \leq \sum_{t=0}^{\infty}((\max\{\boldsymbol{A}_i\})\mathbf{S})^t \tag{2.68}$$

and the infinite sum converges. □

3 Literature Review

In order to pursue a greater understanding on the inner workings of our chosen classifier, we present a literature review on graph-based semi-supervised approaches that attempt to address label noise. This is largely based on the systematic review we conducted (AFONSO, 2020). Within graph-based semi-supervised learning, there are four main strategies which are efficient at reducing the effect of label noise on classification. Section 3.1 presents the overfitting avoidance mechanisms based on eigenfunctions. Then, section 3.2 goes over the self-paced mechanism that encourages learning easy labels first. Following that, we detail a few ℓ_1-norm algorithms in Section 3.3. These are probably the most relevant for our work, as we will be used the benchmarks presented in their papers to compare them to our filter. Another approach that has been shown to be robust to noise is Particle Cooperation and Competition, which is gound in Section 3.4. Lastly, Section 3.5 presents gradient-based approaches that jointly minimize the classification function and the initial labels. Those methods are also a powerful alternative, however they need to store a full propagation matrix, making them unsuitable for our experiments due to memory consumption. Our method `LGC_LVO_Auto` does take inspiration from them, while also offering more scalable performance with respect to unlabelled data.

3.1 Overfitting Avoidance

As we have outlined in Section 2.4, the Laplace-Beltrami operator provides a corresponding eigendecomposition such that every function in $L^2(\mathcal{M})$ can be written as the linear combination of its eigenvectors. If we perform such an eigendecomposition on the graph Laplacian, it yields the set of unit-norm eigenfunctions $(e_i, \ldots, e_n)$, whose respective eigenvalues are $(\lambda_i, \ldots, \lambda_n)$. If we evaluate the smoothness criterion for a given eigenfunction e_i, we have that

$$\mathbf{e}_i^\top \mathbf{L}\mathbf{e}_i = \mathbf{e}_i^\top \lambda_i \mathbf{e}_i = \lambda_i \mathbf{e}_i^\top \mathbf{e}_i = \lambda_i \tag{3.1}$$

As a result, $(e_i, \ldots, e_n)$ is a sequence of eigenfunctions with monotonically nonincreasing smoothness with respect to the graph. Thus, it makes sense to restrict our function to the space of linear combinations of the first few eigenvectors. We say that any algorithm that makes use of this in some manner performs **Smooth Eigenbasis Pursuit (SEP)**.

The idea of restricting our search to the linear combinations of a few smooth eigenfunctions turns out to work very well in practice. The **Laplacian Eigenmaps (LE)** method (BELKIN; NIYOGI, 2003) extracts the first few eigenfunctions of the normalized graph Laplacian $\mathbf{L}_\mathbb{N}$. This can be shown to solve the problem of preserving proximity relations in a lower-dimensional embedding, although some restrictions are necessary to avoid degenerate solutions. In addition, we can optimize the loss function of least squares subject to this restriction. In

Afonso e Berton (2020), we observed that this simple classifier was able to consistently outperform all others for a number of datasets, if the choice of eigenbasis was appropriate. Many times, the ground truth may correspond to a simple hypothesis (i.e. it can be represented with few eigenfunctions), whereas fitting incorrect labels requires a more complex hypothesis. Thus, this can be seen as dealing with label noise via overfitting avoidance.

Any approach based on SEP will optimize for the coefficients of the linear combination of the first m eigenfunctions with nonzero eigenvalues. Specifically, the classification matrix will be defined as

$$\mathbf{F} = \mathbf{U_m} v \tag{3.2}$$

where $\mathbf{U_m} = [e_1\, e_2\, \ldots\, e_m] \in \mathcal{M}^{n \times m}$ is the eigenfunction matrix and $v \in \mathcal{M}_{m \times c}$ encodes the coefficients. Accordingly, the cost is to be minimized with respect to v. In the LE least squares classifier, this is simply

$$Q(v) = \left\| \mathbf{U_m} v - \mathbf{Y} \right\|^2 \tag{3.3}$$

Whenever necessary, the *eigenvalue matrix* may be incorporated into the cost function

$$\forall i \in \{1..m\} : (\mathbf{\Lambda_m})_{ii} = \lambda_i \tag{3.4}$$

Many other classifiers that we will see next will optimize coefficients v, albeit with more intricate cost functions.

The other way to perform overfitting avoidance is to simply increase any parameter which controls the importance of the smoothness term with respect to the graph. This is different than restricting the amount of eigenfunctions: those that are not smooth with respect to the graph will be strongly discouraged within the regularization framework, but never prohibited outright. As such, we can say that the two regularization approaches complement each other. One example of the latter is whenever one purposefully decreases the parameter μ of LGC. In general, this does indeed reduce the reliance of LGC on (possibly) noisy labels, and reduces the effect of label noise (AFONSO; BERTON, 2020).

3.2 Learning Easy Examples First

The *self-paced learning* (SPL) theory is inspired by learning principles of animals and human beings. Much like learning a skill in real life, one must begin with an easy task that gets more challenging as time goes on. As such, the model is trained on easy samples at first, and then moves on to more complex examples (i.e. ones that it is struggling with). The idea is to make the model progressively more mature. Making use of the SPL methodology has the potential to make SSL learning label noise-tolerant, as it is robust to outliers and heavy noise (MENG; ZHAO; JIANG, 2015).

Self-Paced Manifold Regularization (SPMR) (GU; FAN; MENG, 2016) makes the learner operate "at its own pace". The idea is to gradually increase the difficulty of the data

being classified. In practice, samples are linearly weighted with respect to their losses. This is enough to outperform LSSC (see Section 3.3) when artificial label noise is introduced in the following datasets: the MIT CBCL dataset (MIT, 2000), ORL faces (SAMARIA; HARTER, 1994), and USPS handwritten digits (HULL, 1994).

3.3 Using Different Norms for Better Local Adaptativity

Most of the previous label propagation algorithms rely on variants of the same smoothness criterion $\mathbf{F}^\top \mathbf{L}\mathbf{F}$. It can be shown (WANG et al., 2016) that the iterated Laplacian smoother is in fact a kind of ℓ_2-norm:

$$\mathbf{F}^\top \mathbf{L}_{\mathbb{C}}^{(t+1)} \mathbf{F} = \left\| \Delta^{(t+1)} \mathbf{F} \right\|_2^2 \tag{3.5}$$

where $\Delta^{(t+1)}$ is a recursively defined graph difference operator. In particular, $\Delta^{(1)}$ is the weighted, oriented incidence matrix of the graph. One of the worst aspects of using an ℓ_2-norm is that differences in the graph signal cannot to be set to exactly zero. This essentially guarantees that every noisy label will effect the classification function to some extent. In contrast, the ℓ_1-norm minimization provides better *local adaptativity*: some regions may have strong variations, whereas others remain constant.

Ideas from semi-supervised learning are applicable even when all data is labelled. In Agrawal et al. (2013), the LGC cost function was applied along with the restriction $\left\|\mathbf{F}\right\|_0 \leq C$ for some constant C. The reason for this norm is that few positive instances were expected. The minimization of this cost provided soft labels to the final classifier, a multi-label regression forest, instead of hard labels. This enabled the classifier to not treat every label with the same certainty, reducing the effect of label noise.

Large-Scale sparse coding (LSSC) (LU et al., 2015) uses the ℓ_1-norm to transform noise-robust semi-supervised learning into a generalized sparse coding problem. Although not the first to use this type of norm, they manage to make it scalable by providing a large-scale extension. This extension is based on Fergus, Weiss e Torralba (2009) and involves the eigenfunctions of the normalized Laplacian matrix $\mathbf{L}_{\mathbb{N}}$. Specifically, they restrict $\mathbf{F} = \mathbf{U}_{\mathbf{L}_{\mathbb{N}};m}\mathbf{v}$ where $\mathbf{U}_{\mathbf{L}_{\mathbb{N}};m}$ is an $n \times m$ matrix whose columns are the m eigenvectors with smallest eigenvalues. Thus, LSSC performs smooth eigenbasis pursuit. This results in a cost

$$Q(\mathbf{v}) = \frac{1}{2}\left\|\mathbf{U}_{\mathbf{L}_{\mathbb{N}};m}\mathbf{v} - \mathbf{Y}\right\|_2^2 + \lambda \left\|\mathbf{\Lambda_m}^{\frac{1}{2}}\mathbf{v}\right\|_1 \tag{3.6}$$

Semi-supervised learning with noise can also be seen as a graph-signal restoration problem, as in Mao et al. (2016), which uses a *generalized graph smoothness prior* to learn an image classifier given noisy labels. The objective includes a fidelity term to minimize the ℓ_0-norm between the observed labels and a reconstructed graph-signal. By assuming that both the signal $\mathbf{F}$

and its gradient are smooth, the cost is

$$Q(\mathbf{F}) = \left\|\mathbf{Y}_{\mathcal{L}} - \mathbf{F}_{\mathcal{L}}\right\|_0 + \lambda_1 \mathbf{F}^\top \mathbf{L}\mathbf{F} + \lambda_2 \mathbf{F}^\top \mathbf{L}^2 \mathbf{F} \tag{3.7}$$

Thus we have two smoothness terms, whereas the first term is an ℓ_0-norm that is approximated in practice by

$$(\mathbf{Y} - \mathbf{DF})^\top \mathbf{A}(\mathbf{Y} - \mathbf{DF}) \tag{3.8}$$

where $\mathbf{A}$ is a diagonal matrix containing weights that are set and updated so that the ℓ_2 norm mimics the ℓ_0 norm. This is accomplished by means of an *iterative reweighted least squares strategy* (IRLS) (DAUBECHIES et al., 2010).

Semi-Supervised learning under Inadequate and Incorrect supervision (SIIS) (GONG et al., 2017) applies to the unnormalized Laplacian $\mathbf{L}$ ideas similar to the ones in LSSC. Let $\mathbf{U}$ be the matrix containing the eigenvectors. SIIS is yet another method that uses *smooth eigenbasis pursuit*. It restricts the classifying function to be a combination of the m smoothest eigenfunctions of the graph Laplacian. To do this, the matrix of eigenfunctions $\mathbf{U}$ is replaced with $\mathbf{U}_m$, which contains only the m eigenfunctions with smallest eigenvectors. The corresponding diagonal matrix of eigenvalues is $\mathbf{\Lambda_m}$. Both the first and second term are regularized with an $\ell_{2,1}$ norm. Additionally, there is a third term to further encourage use of the smoothest eigenfunctions, even when already restricted to the m most smooth. We end up with:

$$Q(\mathbf{v}) = \left\|\Delta \mathbf{U_m v}\right\|_{2,1} + \lambda_1 \left\|(\mathbf{U_m v})_\mathbf{l} - \mathbf{Y}_{\mathcal{L}}\right\|_{2,1} + \lambda_2 tr(\mathbf{v}^\top \mathbf{\Lambda_m v}) \tag{3.9}$$

Notably, SIIS was shown to be superior to LSSC and SPMR (see Sections 3.3, 3.2) on datasets like ISOLET and RCV1 (LEWIS et al., 2004) under heavy label noise. That being said, the algorithm has some parameter sensitivity, and in practice seemed to favour the fitting term.

3.4 Particle Cooperation and Competition

For `LGC_LVO_Auto`, we minimized both loss functions with the use of Tensorflow. The hcetition is one based on **dominating vertices**. Notably, it is an iterative process where, at each step, a particle may perform one of the following: a **random walk**, which is the same as using the probability transition matrix $\widetilde{W} = \mathbf{D}^{-1}\mathbf{W}$ to select the next random node; or a **deterministic walk** (also known as **greedy movement**), where the particle prioritizes moving to a neighbour if it is close to the particle's home node, and also according to the **domination level** of its team on that neighbour. The domination level measures how much a given team has taken control over some vertex, and it is increased or decreases over time depending on the presence of particles in it.

The original formulation of PCC was shown to be robust to label noise (BREVE; ZHAO; QUILES, 2010), however it was also subject to a territory switch phenomenon when label noise was high enough, which decreased accuracy. This was fixed in a subsequent version (BREVE; ZHAO; QUILES, 2015).

3.5 Jointly Optimizing Labels and Prediction

So far, we have seen many methods employ overfitting avoidance, self-paced learning and ℓ_1-norm regularization to effectively reduce the overall effect of noisy labels on classification. However, the noisy labels are addressed only indirectly, as they are not explicitly corrected. In contrast, a **bivariate formulation** tries to jointly optimize the label matrix $\mathbf{Y}$, as well as the classification matrix $\mathbf{Y}$.

A first example of a bivariate formulation is *Approximate kNN-SGSSL with noisy label handling* (TANG et al., 2011) (AkNN-SGSSL_dn). It also the uses ℓ_1-norm on both graph construction and cost minimization. To improve the effficiency of graph construction, it sparsely reconstructs each sample from its k nearest neighbours in feature space instead of using all the other samples. From this, we get a sparse affinity matrix $\mathbf{W}$. For the cost minimization, the smoothing term is a bit different than the usual Laplacian, as it uses reconstruction error:

$$\mathbf{F}^\top L_{rec}\mathbf{F} = \left\|\mathbf{F} - \mathbf{W}\mathbf{f}\right\|^2 \tag{3.10}$$

$$= \sum_{i=1}^{N}\left\|\mathbf{F}_i - \sum_{j\neq i}\mathbf{W}_{ij}\mathbf{F}_j\right\|^2 \tag{3.11}$$

where $L_{rec} = (\mathbf{I}-\mathbf{W})^\top(\mathbf{I}-\mathbf{W})$ is equal to the Laplacian L only if we first row-normalize $\mathbf{W}$. Consequently, the smoothness term is actually an ℓ_2-norm. With that being said, the ℓ_1-norm does make an appearance, namely in the fitting term. The bivariate cost formulated is given by

$$Q(\mathbf{F},\mathbf{Y}) = \mathbf{F}^\top L_{rec}\mathbf{F} + \lambda_1\left\|\mathbf{F}-\mathbf{Y}\right\|^2 + \lambda_2\left\|\mathbf{Y}-\mathbf{Y}_0\right\|_1 \tag{3.12}$$

where $\mathbf{Y}_0$ is the noisy, initial value of $\mathbf{Y}$. The use of the *generalized minimal residual method* (GMRES) (SAAD; SCHULTZ, 1986) speeds up the minimization process considerably.

3.5.1 GTAM: the Alternating Minimization

The *Graph Transduction via Alternating Maximization* (GTAM) (WANG; JEBARA; CHANG, 2008) can be described as a bivariate, greedy gradient-based formulation built upon the LGC classifier. The influence of LGC is immediately apparent by looking at the cost function. But first, we must introduce the *normalized label matrix* $\widetilde{\mathbf{Y}}$:

$$\widetilde{\mathbf{Y}} = \mathbf{V}\mathbf{Y} \tag{3.13}$$

where $\mathbf{V} = v(\mathbf{Y}) \in \mathbb{R}^{n\times n}$ is a diagonal matrix which depends on $\mathbf{Y}$:

$$\forall i \in \{1..n\} : \mathbf{V}_{ii} = \sum_{j=1}^{c}\mathbb{1}(\mathbf{Y}_{ij}=1)\left(\omega_j\frac{\mathbf{D}_{ii}}{\sum\limits_{k:\mathbf{Y}_{kj}=1}\mathbf{D}_{kk}}\right) \tag{3.14}$$

here, $\mathbb{1}(\cdot)$ is an indicator function which yields 1 if the statement is true, and zero otherwise. Also, ω_j is the prior class probability for a given class j. The normalized label matrix encodes

the prior belief that labels from vertices with high degree should propagate more intensely through the graph. Its columns will have their sum equal to the respective prior class probabilities, which is later useful for stabilizing the minimization procedure. The cost function is the same as the one used by LGC, except that it uses the label matrix as another variable:

$$\mathcal{Q}(\mathbf{F}, \mathbf{Y}) = \frac{1}{2} tr\left(\mathbf{F}^\top \mathbf{L}\mathbf{F} + (\mathbf{F} - \widetilde{\mathbf{Y}})^\top \mathbf{\Sigma}(\mathbf{F} - \widetilde{\mathbf{Y}})\right) \tag{3.15}$$

It must be noted that, within our literature review, the convention is to have $\mathbf{\Sigma} = \mu\mathbf{I}_{\mathbf{n}\times\mathbf{n}}$. Still, we choose to use the more general Σ, which provides a useful generalization for GTAM, much like the one for LGC in Sousa (2017).

At first glance, this cost function is ill-posed, as setting $\mathbf{F}$ and $\widetilde{\mathbf{Y}}$ to zero yields a solution. Unlike the cost in AkNN-SGSSL_dn (Equation 3.12), it does not discourage changes to the initial label matrix. This needs to be addressed. The way GTAM works is by forcing the initial labels to stay the same. For any integer i, let $\mathbf{Y}_i$ denote the label matrix after i iterations, and $\widetilde{\mathbf{Y}}_i$ its normalized version. We will also use the subscript $\mathcal{L}_i$ to denote the label set and $\mathcal{U}_i$ its complement, after i iterations. GTAM's cost function is more accurately described as

$$\begin{aligned} &\mathcal{Q}(\mathbf{F}, \mathbf{Y}) = \frac{1}{2} tr\left(\mathbf{F}^\top \mathbf{L}\mathbf{F} + (\mathbf{F} - \widetilde{\mathbf{Y}})^\top \mathbf{\Sigma}(\mathbf{F} - \widetilde{\mathbf{Y}})\right) \\ &\text{such that } \left(\mathbf{Y} \in \mathbb{B}_{n\times c} \wedge \mathbf{Y}\mathbf{1}_c = \mathbf{1}_n \wedge \mathbf{Y}_{\mathcal{L}_0} = \mathbf{Y}_{0_{\mathcal{L}_0}}\right) \end{aligned} \tag{3.16}$$

One might say that there is a major flaw in GTAM: the noisy labels are never actually corrected in the label matrix, due to being fixed at the beginning. Yet, experimental results (WANG; JEBARA; CHANG, 2008) show that GTAM is superior to LGC on a variety of experimental settings. This is either because of label normalization, or because GTAM's optimization works around this issue by adding labels closer to the ones that are less likely to be noisy.

The name given to GTAM alludes to the fact that it uses an alternating minimization procedure to arrive at its solution. This alternating minimization is a *greedy* approach based on the *gradient* of the cost (Equation 3.16). In Section 2.5.4, we showed that $\mathbf{F}$ has a closed-form solution, given $\mathbf{Y}$ (Equation 2.56). On the other hand, solving Equation 3.16 is a problem which is NP, as it is equivalent to solving a linearly constrained binary integer programming problem (KARP, 1972). By substituting the optimal $\mathbf{F}$ into Equation 3.16, we get

$$Q(\mathbf{F}, \mathbf{Y}) = \frac{1}{2} tr\left(\widetilde{\mathbf{Y}}^\top \mathbf{A}\widetilde{\mathbf{Y}}\right) \tag{3.17}$$

where

$$\mathbf{A} = \widehat{\mathbf{P}}^\top \mathbf{L}\widehat{\mathbf{P}} + (\widehat{\mathbf{P}}^\top - \mathbf{I}_{\mathbf{n}\times\mathbf{n}})(\widehat{\mathbf{P}} - \mathbf{I}_{\mathbf{n}\times\mathbf{n}}) \tag{3.18}$$

$$\widehat{\mathbf{P}} = \mathbf{\Sigma}\mathbf{\Theta} \tag{3.19}$$

The gradient of this reformulated cost is

$$\frac{\partial \mathcal{Q}}{\partial \widetilde{\mathbf{Y}}} = \frac{1}{2}(\mathbf{A} + \mathbf{A}^\top)\widetilde{\mathbf{Y}} \tag{3.20}$$

In general, the affinity matrix $\mathbf{W}$ is assumed to be symmetric, which also implies $\widehat{\mathbf{P}} = \widehat{\mathbf{P}}^\top$ and $\mathbf{A} = \mathbf{A}^\top$. The optimal *labelling operation* at iteration t is calculated as:

$$(i^+, j^+) = \underset{\{(i,j):(i\in\mathcal{U})\wedge(1\leq j\leq c)\}}{\arg\min} \left(\frac{\partial \mathcal{Q}}{\partial \widetilde{\mathbf{Y}}}\right)_{ij} \tag{3.21}$$

$$\mathcal{L} = \mathcal{L} \cup \{i^+\} \tag{3.22}$$

$$\mathbf{Y}_{i^+j^+} = 1 \tag{3.23}$$

This may continue until all instances are labelled, or it may be ended early so that it is less computationally expensive.

3.5.2 LDST: Label Diagnosis through Self-tuning

Label Diagnosis through Self-Tuning (WANG; JIANG; CHANG, 2009) is an extension of the GTAM approach that uses both directions of the gradient. In addition to GTAM's labelling operation, the first s iterations also include an *unlabelling operation*, which selects the currently labelled instance which would decrease the cost function the most.

$$(i^-, j^-) = \underset{\{(i,j):(\mathbf{Y}_{ij}=1)\}}{\arg\min} \left(\frac{\partial \mathcal{Q}}{\partial \widetilde{\mathbf{Y}}}\right)_{ij} \tag{3.24}$$

$$(i^+, j^+) = \underset{\{(i,j):(i\in\mathcal{U})\wedge(1\leq j\leq c)\}}{\arg\min} \left(\frac{\partial \mathcal{Q}}{\partial \widetilde{\mathbf{Y}}}\right)_{ij} \tag{3.25}$$

$$\mathcal{L} = (\mathcal{L} \cup \{i^+\}) \setminus \{i^-\} \tag{3.26}$$

$$\mathbf{Y}_{i^-j^-} = 0 \tag{3.27}$$

$$\mathbf{Y}_{i^+j^+} = 1 \tag{3.28}$$

As such, one can argue that it approximately minimizes the following cost

$$\mathcal{Q}(\mathbf{F}, \mathbf{Y}) = \frac{1}{2} tr\left(\mathbf{F}^\top \mathbf{L}\mathbf{F} + (\mathbf{F} - \widetilde{\mathbf{Y}})^\top \mathbf{\Sigma}(\mathbf{F} - \widetilde{\mathbf{Y}})\right)$$
$$\text{such that } \left(\mathbf{Y} \in \mathbb{B}_{n\times c} \wedge \mathbf{Y}\mathbf{1}_c = \mathbf{1}_n \wedge \left\|\mathbf{Y}_{\mathcal{L}_0} - \mathbf{Y}_{0_{\mathcal{L}_0}}\right\|_0 \leq s\right) \tag{3.29}$$

In Wang, Jiang e Chang (2009), LDST is shown to be superior to GTAM on a toy dataset but also for the task of web image search on a set of images manually crawled from the photo sharing website Flickr.

3.5.3 Alternating minimization as a Greedy Max Cut

In Wang, Jebara e Chang (2013), the two-class version of GTAM is reduced to a linearly constrained weighted max-cut problem over the graph $G_{\mathbf{A}} = (\mathbf{X}, \mathbf{A})$. We start by rewriting Equation 3.17:

$$Q(\mathbf{Y}) = \frac{1}{2} tr\left(\widetilde{\mathbf{Y}}^\top \mathbf{A}\widetilde{\mathbf{Y}}\right) = \frac{1}{2} tr\left(\mathbf{A}\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^\top\right) = \frac{1}{2} tr\left(\mathbf{A}\mathbf{R}\right) \tag{3.30}$$

where $\mathbf{R} = \widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^\top$. The constraint $\mathbf{Y}\mathbf{1}_c = \mathbf{1}_n$ is satisfied if and only if

$$\mathbf{Y} = [\mathbf{y}, \mathbf{1}_n - \mathbf{y}] \tag{3.31}$$

The problem is then restricted to the binary domain by having $\forall i \in [1..n] : y_i \in \{0, 1\}$. By substituting (3.31) in (3.30), we have

$$Q(\mathbf{y}) = \frac{1}{2} tr\left(\mathbf{A}\mathbf{1}_n\mathbf{1}_n^\top - \mathbf{A}\mathbf{y}(\mathbf{1}_n - \mathbf{Y})^\top - \mathbf{A}(\mathbf{1}_n - \mathbf{Y})\mathbf{y}\right) \tag{3.32}$$

The first term is ignored because it doesn't depend on $\mathbf{Y}$. Using the fact that $\mathbf{A} = \mathbf{A}^\top$ and the properties of the trace of a matrix, the optimal value for $\mathbf{y}$ is determined as

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} Q(\mathbf{y}) = \arg\max_{\mathbf{y}} \mathbf{y}^\top\mathbf{A}(\mathbf{1}_n - \mathbf{y}) \tag{3.33}$$

Using Lemma 2.4.1, this is equivalent to

$$\mathbf{y}^* = \sum_{1\leq i,j\leq n} \mathbf{A}_{ij} \cdot \mathbf{y}_i(1 - \mathbf{y}_j) \sum_{1\leq i,j\leq n} \mathbf{A}_{ij} \cdot \mathbf{y}_i(1 - \mathbf{y}_j) = \sum_{i\in S_0, j\in S_1} \mathbf{A}_{ij} \tag{3.34}$$

where $S_0 = \{i \mid \mathbf{y}_i = 0\}, S_1 = \{i \mid \mathbf{y}_i = 1\}$. Since $\mathbf{X}_{S_0\cup S_1} = \mathbf{X}$ and $\mathbf{X}_{S_0\cap S_1} = \emptyset$, this is equivalent to the max-cut problem on the graph $G_{\mathbf{A}}$. Unfortunately, calculating a solution to the max-cut problem is difficult when the entries are not necessarily non-negative, as is the case for matrix $\mathbf{A}$.

By looking at Equation 3.34, we see that the diagonals of $\mathbf{A}$ do not alter the solution, and may be assumed to be zero. It is also shown (WANG; JEBARA; CHANG, 2013) that the multi-class version of GTAM is equivalent to the maximum K-cut problem.

4 Proposal

This chapter is dedicated to presenting our proposal. We begin by listing out the objectives we have sought to accomplish in Section 4.1. Then, we present our novel approach to an LGC filter in Section 4.2. Finally, we go into detail about our methodology to validate our filter in Section 4.3. This includes: the programming language and packages; the datasets which will serve as our benchmark; our general workflow.

4.1 Objectives

The main objective of this work is

- To build upon our recently developed graph-based semi-supervised learning filter by improving the stopping criteria selection or removing it outright.

This main objective encompasses a number of more specific goals that have driven this work.

- Develop an approach for letting the user set the stopping criteria in a way that is more useful than simply using a fixed number of removed labels.
- Take one step further, and try to eliminate the need for parameter selection entirely.
- Evaluate the ensuing filter, embedded in its LGC baseline, and compare with previously reported results of ℓ_1-norm methods.
- Evaluate whether using the filter beforehand adds any robustness to the LGC classifier, or if it just uses LGC's own inherent robustness to correct labels without improving classification.

4.2 Our Novel Approach to an LGC Filter Without a Manual Stopping Criteria

In this section, we discuss an inherent problem of the LGC algorithm, and propose to fix it by formulating a new optimization problem. The problem is shown to be convex. Finally, we provide an optimization procedure based on gradient descent.

4.2.1 Issues within the Cost Function of LGC

Recall the LGC cost function (Equation 2.29). We will take a look at its solution, argue that it has some undesirable properties, and introduce a new semi-supervised classifier that automatically picks the best α parameter.

The LGC solution, $\mathbf{F} = \mathbf{PY}$, relies on a propagation matrix $\mathbf{P}$, whose entries are related to a random walk (Equation 2.43). Let vertex i be the starting point of our random walk. The row-normalized affinity matrix encodes transition probabilities. At any step t, the estimated contribution of whichever vertex we are visiting (say, j) is boosted with a reward proportional to α^t. As the random walk goes on indefinitely, the same vertex j may be visited again and again. The total reward is finite, however, because of the exponential decay. More precisely, $\mathbf{P}_{ij}$ represents the expected value of the total reward yielded to vertex j during a random walk that started on vertex i.

There is one major problem with LGC's solution: the diagonal of $\mathbf{P}$. At first glance, we would think that "fitting the labels " means looking for a model that explains our data very well. In reality, this translates to memorizing the labelled set. The main problem resides within the **diagonal** of the propagation matrix. Any entry $\mathbf{P}_{ii}$ stores the **self-influence** of a vertex, which is calculated according to the expected reward obtained by looping around and visiting itself. The optimal solution w.r.t. label fitting occurs when α tends to zero. For labelled instances, an initial reward is given for the starting vertex itself, and the remaining are essentially ignored.

We argue that the diagonal is directly related to **overfitting**. It essentially tells the model to rely on the label information it knows. There are a few analogies to be made: say that we are optimizing the number of neighbours k for a KNN classifier. The analogue of "LGC-style optimal label fitting" would be to include each labelled instance as a neighbour to itself, and set $k = 1$. This is obviously not a good criteria. The answer that maximizes a *proper* "label fitting" criteria, in this case, is selecting k that minimizes classification error with the extremely important caveat: no cheating, i.e. using each instance's own label is prohibited. The optimal parameter k with respect to this criteria becomes data-dependent, unlike the previous one.

The problem of **diagonal dominance** is also observed elsewhere, namely for support vector machines. According to Greene e Cunningham (2006), it is more likely to occur when the data is sparse and high-dimensional. When an SVM uses a diagonally dominant kernel matrix, it memorizes the training data, leading to severe overfitting. In practice, these kernels suffer from poor generalization. It is worthwhile to note that getting rid or minimizing the effect of the diagonal is much harder for SVMs, as the kernel matrix is required to still be positive definite. This does not come as a surprise, given that any classifier that eliminates this dependence entirely can become optimal with respect to **leave-one-out validation error**. This is an unrealistic goal for SVMs and neural networks, but feasible for us.

In spite of the problems we have presented, the family of LGC solutions remains very

interesting to consider. We will have to eliminate diagonals, however. Let

$$\mathbf{H}(\alpha) := \left((\mathbf{I} - \alpha\mathbf{S})^{-\mathbf{1}} - diag((\mathbf{I} - \alpha\mathbf{S})^{-1})\right)\mathbf{Y} \tag{4.1}$$

If we also row-normalize (a small constant ϵ may be added for stabilization), we end up with

$$\widehat{\mathbf{H}}(\alpha) := (diag(\epsilon\mathbf{1}_{n\times n} + \mathbf{H}(\alpha))^{-1}\,\mathbf{H}(\alpha) \tag{4.2}$$

We previously stated that the minimization of the LGC cost should correspond with a trade-off between **graph smoothness** and **label fitting**. The criterion we will use for our algorithm appears familiar at first glance, and is stated as follows:

> "Our algorithm should seek an hypothesis that is able to *explain the data well* so that each label can be *recovered* from the remaining ones.

The objective is making sure that the label's initial information is not used for determining its final label. If we only consider this part, a suitable cost is:

$$C(\alpha) = \left(\left\|\widehat{\mathbf{H}}(\alpha)_{\mathcal{L}} - \mathbf{Y}_{\mathcal{L}}\right\|^2\right) \tag{4.3}$$

We can also use $\mathbf{H}(\alpha)$ instead of $\widehat{\mathbf{H}}(\alpha)$ here. The downside is, it doesn't output probabilities, which may have a significant effect (AFONSO; BERTON, 2020).

One final observation here is that we can extend our framework to the generalized version of LGC (see Equation 2.56), which leads to

$$\mathbf{H}(\mathbf{\Sigma}) := \left((\mathbf{L} + \mathbf{\Sigma})^{-1} - diag((\mathbf{L} + \mathbf{\Sigma})^{-1}\right)\mathbf{\Sigma}\mathbf{Y} \tag{4.4}$$

This is a dangerous proposition, however. Leave-One-Out validation is usually reliable, therefore optimizing a single parameter α is unlikely to overfit the data. But we must be aware of what we are doing. This could be addressed if we separate some labels for evaluating this optimization process. Another way is to simply restrict the amount of free parameters. If we optimize the entire diagonal matrix $\mathbf{\Sigma}$, we could conceivably create a path with high diffusion rate connecting labels of the same class, and zero diffusion everywhere else. We may then look to force smoothness of $\mathbf{\Sigma}$ w.r.t. the graph, or force diffusion rates to be equal for each class.

4.2.2 Minimization of LGC's LOO Error with respect to α

In this subsection, we will calculate the gradient of the leave-one-out error of LGC's predictions, with respect to α. We will also go over why minimizing the cost this way might be unreasonable, which will lead us to the approach proposed in this work.

Once again, the LGC cost is

$$\mathcal{Q}(F) = \frac{1}{2}\left(tr(F^\top \mathbf{L}_{\mathbb{N}} F) + \frac{1-\alpha}{\alpha}\left\|F - Y\right\|^2\right) \tag{4.5}$$

The solution is

$$\mathbf{F} = \mathbf{PY} \tag{4.6}$$

for

$$\mathbf{P} = (\mathbf{I} - \alpha \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}})^{-1} \in \mathbb{R}^{n \times n} \tag{4.7}$$

An equivalent solution would be

$$\mathbf{F} = \widehat{\mathbf{P}}_{\mathcal{L}} \mathbf{Y}_{\mathcal{L}} \tag{4.8}$$

where $\widehat{\mathbf{P}}$ is the result of row-normalizing $\mathbf{P}$ after adding some small value ϵ and zeroing out the diagonal. In other words, if l is the number of labels and c the number of classes, our classification matrix is expressed as

$$\forall i \in \{1..l\}, j \in \{1..c\} : \mathbf{F}_{ij} = \frac{\sum_{k=0}^{l} (\mathbf{P}_{ik} + \epsilon) \mathbb{1}_{[Y_{kj}==1 \wedge k \neq i]}}{\sum_{k=0}^{l} (\mathbf{P}_{ik} + \epsilon) \mathbb{1}_{[k \neq i]}} \tag{4.9}$$

So

$$\frac{\partial \left\| \mathbf{F}_{\mathcal{L}} - \mathbf{Y}_{\mathcal{L}} \right\|^2}{\partial \alpha} = 2(\mathbf{F}_{\mathcal{L}} - \mathbf{Y}_{\mathcal{L}}) \circ \frac{\partial \mathbf{F}_{\mathcal{L}}}{\partial \alpha} \tag{4.10}$$

where $\circ$ denotes the Hadamard product, that is, elementwise multiplication. Let us define $\mathbf{F}_{ij}^{num}$,$\mathbf{F}_{ij}^{den}$ as the numerator and denominator associated with Equation 4.9, respectively. Then,

$$\frac{\partial \mathbf{F}_{ij}}{\partial \alpha} = \frac{(\mathbf{F}_{ij}^{num})'(\mathbf{F}_{ij}^{den}) - (\mathbf{F}_{ij}^{den})'(\mathbf{F}_{ij}^{num})}{(\mathbf{F}_{ij}^{den})^2} \tag{4.11}$$

where $'$ is also used to denote the partial derivative w.r.t. to α. Also

$$(\mathbf{F}_{ij}^{num})' = \sum_{k=0}^{l} (\mathbf{P}'_{ik}) \mathbb{1}_{[Y_{kj}==1 \wedge k \neq i]} \tag{4.12}$$

$$(\mathbf{F}_{ij}^{den})' = \sum_{k=0} (\mathbf{P}'_{ik}) \mathbb{1}_{[k \neq i]} \tag{4.13}$$

As discussed before, P is the limit of the iterative procedure. The closed expression for F at any iteration is

$$F(t) = (\alpha S)^{t-1} Y + (1 - \alpha) \sum_{i=0}^{t-1} (\alpha S)^i Y \tag{4.14}$$

The formula for the derivative of the inverse matrix is given as

$$\frac{\partial \mathbf{P}}{\partial \alpha} = -\mathbf{P} \frac{\partial (\mathbf{I} - \alpha \mathbf{S})}{\partial \alpha} \mathbf{P} = \mathbf{PSP} \tag{4.15}$$

As we only need the *labelled versus labelled* portion of this gradient, this simplifies to

$$(\mathbf{P}_{\mathcal{L}})^{\top} \mathbf{S} \mathbf{P}_{\mathcal{L}} \tag{4.16}$$

where $\mathbf{P}_{\mathcal{L}}$ consists of the $n \times l$ submatrix containing the columns corresponding to labelled instances.

The biggest problem with this approach is that it requires computing a subset of the propagation matrix for each gradient step. This would lead to a very slow optimization process, even if we use approximation techniques. This approach may be worth investigating in the future, but a lot of work will be needed to make it desirable in practice. Therefore, we will instead try to minimize leave-one-out error **with respect to the initial labels**. This will turn out to be much more efficient.

4.2.3 An Overview of our Filter `LGC_LVOf`

An approach that yields a classification and a corrected label matrix was developed during the author's master degree (AFONSO, 2020), named *Local and Global Consistency Leave-One-Out filter (`LGC_LVOf`)*. The main idea here is that we assume that the propagation model of LGC is correct: that is, a high $\mathbf{P}_{ij}$ correctly describes that instances $\mathbf{x}_i$ gets label information from $\mathbf{x}_j$. More generally, the propagation matrix $\mathbf{P}$ correctly describes the label of each instance as a linear combination of the other available labels. Crucially, we assume that there should be *no contradictions*. Assume that, initially, $\mathbf{Y}_{ij} = 1$. We want to know if instance i would still be assigned to class j if $\mathbf{F}_{ij}$ was completely determined from the other labels, i.e. if we had removed its label (and no other labels) from the label propagation procedure. Notably, we can get the result of the modified label propagation by setting the diagonal of $\boldsymbol{P}$ to zero, eliminating any influence a label has on itself:

$$\mathbf{F} = (\mathbf{P} - diag(\mathbf{P}))\mathbf{Y} + \epsilon \mathbf{1}^{\mathbf{n}\times\mathbf{c}} \tag{4.17}$$

where $diag(\mathbf{P})$ contains the diagonal entries of the propagation matrix, and ϵ is a small constant to ensure that the sum of each row is positive. To evaluate contradictions, we only need the result of $\mathbf{F}$ on labelled instances:

$$\mathbf{F}_{\mathcal{L}} = (\mathbf{P}_{\mathcal{L}\mathcal{L}} - diag(\mathbf{P}_{\mathcal{L}\mathcal{L}}))\mathbf{Y}_{\mathcal{L}} + \epsilon \mathbf{1}_{\mathbf{l}\times\mathbf{c}} \tag{4.18}$$

`LGC_LVOf` selects the next most likely noisy instance by evaluating

$$\underset{(i,j):(\mathbf{Y}_{\mathcal{L}})_{ij}=1}{\arg\min} \; \mathbf{Y}_{\mathcal{L}} - (deg(\mathbf{F})^{-1}\mathbf{F}) \tag{4.19}$$

where the deg function yields a diagonal matrix containing the sum of each row. When labels are few (as is usually the case in SSL), `LGC_LVOf` benefits greatly from only computing a small subset of the propagation matrix $\mathbf{P}$. Moreover, $\mathbf{F}_{\mathcal{L}}$ can be updated efficiently (AFONSO, 2020): once the propagation matrix is calculated, the filter will perform $\mathcal{O}(cl)$ operations to detect another noisy label.

4.2.3.1 Calculating only a Subset of the Propagation Matrix

The desired submatrix $\mathbf{P}_{\mathcal{LL}}$ can be computed similarly to power iteration (Equation 2.41), with the difference being that we substitute $\mathbf{Y}$ by the following matrix:

$$\widetilde{I} = \begin{bmatrix} I_{l\times l} \\ \mathbf{0}_{(n-l)\times l} \end{bmatrix} \tag{4.20}$$

To be more efficient, we must take advantage of the sparsity of $\mathbf{W}$. Assume that we use a KNN neighbourhood with k neighbours. In AFONSO e Berton (2020), we showed that the approximation of the submatrix can be accomplished with a time complexity of $\mathcal{O}(tcknl)$, where t denotes the number of operations. Usually, the matrix is only guaranteed to converge when $t = n$. In practice, using a much lower t yielded good results. Without delving into advanced math, we posit that that the length of a random walk that yields a good result likely decreases proportionally to:

- The number of classes c, if classes are balanced and with little overlap. There is no need to propagate label information to far away instances corresponding exclusively to other classes.
- The number of labels l. When the number of labels increases, a noisy label is more likely to contradict something in its vicinity.

To further optimize for scalability, there is also the possibility of using an Anchor Graph (LIU; WANG; CHANG, 2012), with complexity $\mathcal{O}(m^3)$ for a set number of anchors m.

4.2.4 Improving our Filter with a Threshold Approach

In Afonso (2020), we introduced our `LGC_LVOf` filter, which measures leave-one-out consistency with the predictions of the LGC model. Originally, this method required the user to simply specify the number of removed labels r. This is less than ideal for the end user, given that this quantity is unknown beforehand. There are two ways to remedy this: one is, of course, `LGC_LVO_Auto` (Section 4.2.5), which is the main focus of this work. But, automatic label correction does have an inherent downside. As `LGC_LVO_Auto` optimizes labels on its own, the user loses some control. That is, he cannot control the intensity of this filter. Therefore, **another approach developed during this work was to provide a way for the end user to make an informed decision regarding the intensity of the filter**. Clearly, making the user specify the exact number of labels to be removed is not enough. As a result, we have developed a new threshold approach, described by Algorithm 1. We calculate the **consistency of each instance with respect to the LGC model**, which is retrieved during the execution of `LGC_LVOf`. This value simply measures how little confidence the label of instance i^* had in its original class j^* at the moment it was selected to be removed. As one can observe in Algorithm 1, this is equal to

Algorithm 1 Leave-one-out filter for the LGC algorithm (LGC_LVOf), with THRESHOLD information (AFONSO, 2020)

Input:
Initial binary label matrix $Y_{\mathcal{L}} \in \mathbb{R}^{l \times c}(\{0,1\})$;
Propagation submatrix $P_{\mathcal{LL}} \in^{\mathbb{R}} l \times l(\mathbb{R})$;
Number of labeled instances l
Output:
`noisyIndexes`: Indices of identified noisy labels.
`suggestedClasses`: Suggested classes for noisy labels .

procedure LGC_LVOF($\mathbf{Y}_{\mathcal{L}}, \mathbf{P}_{\mathcal{LL}}$)
  `labeledIndexes` $\leftarrow \{1 \dots l\}$
  for $i \in$ `labeledIndexes` **do**
    $(P_{\mathcal{LL}})_{ii} \leftarrow 0$ ▷ remove label self-influence
  end for
  $F \leftarrow P_{\mathcal{LL}} Y_{\mathcal{L}}$
  `Q_values` $\leftarrow \{\}$;`noisyIndexes` $\leftarrow \{\}$; `suggestedClasses` $\leftarrow \{\}$
  while Stopping criteria not reached **do**
    $Q \leftarrow$ `to_class_probabilities`$(F, Y_{\mathcal{L}}) - Y_{\mathcal{L}}$
    $(i^*, j^*) \leftarrow argmin_{(i:i \in \texttt{labeledIndexes}; j:1 \leq j \leq c)} Q_{ij}$
    $k^* \leftarrow argmax_{(k:1 \leq k \leq c)} Q_{i^*k}$
    Remove i^* from `labeledIndexes`
    Add i^* to `noisyIndexes`
    Add k^* to `suggestedClasses`
    Add $1 - Q_{i^*j^*}$ to `Q_values`
    $F_{[:,j*]} \leftarrow F_{[:,j^*]} - P_{\mathcal{LL}[:,i^*]}$ ▷ remove contribution
  end while
  return `noisyIndexes`,`suggestedClasses`,`Q_values`
end procedure

$1 - Q_{i^*j^*}$. Those values are then stored in the list `Q_values`. When we are using this approach, our stopping criteria is reached whenever this confidence value exceeds the threshold specified by the user. Alternatively, we may execute until no label is left, and then the user makes his decision by looking at the plot of confidence values.

4.2.5 Automatic Correction of Noisy Labels based on the LGC Leave-One-Out Filter

The major drawback of the `LGC_LVOf`, as presented in Afonso (2020) is that it needs an extra parameter r, which is the number of labels to remove. The optimal r is usually around the number of noisy labels, which is unknown to us. This is a significant disadvantage of the original `LGC_LVO` formulation. This was somewhat addressed in AFONSO e Berton (2020): we can instead use a threshold, which tells us how much labels can deviate from the original model. Nonetheless, it is desirable to solve this problem in a way that removes such a parameter. We will do this by introducing a new optimization problem.

Once again, assume that the modified LGC solution is given by $\widetilde{\mathbf{P}}\mathbf{Y}_{\mathcal{L}}$, where

$$\widetilde{\mathbf{P}} := (\mathbf{P}_{\mathcal{L}\mathcal{L}} - diag(\mathbf{P}_{\mathcal{L}\mathcal{L}})) \tag{4.21}$$

Our optimization problem is to optimize a matrix $\mathbf{\Omega}$ indicating the updated reliability of each label:

$$\min_{\mathbf{\Omega}} \left\| deg(\widetilde{\mathbf{P}}\mathbf{\Omega}\mathbf{Y}_{\mathcal{L}})^{-1}\widetilde{\mathbf{P}}\mathbf{\Omega}\mathbf{Y}_{\mathcal{L}} - \mathbf{Y}_{\mathcal{L}} \right\|^2 \qquad \text{such that} \quad \mathbf{0}_{l\times l} \leq \mathbf{\Omega} \tag{4.22}$$

In this cost, $\mathbf{\Omega}$ is a **diagonal matrix** whose entries encode the **confidence** of each label. This is consistent with our original `LGC_LVOf`, which opts for removing labels instead of relabelling. We insist on degree normalization, i.e. converting each row to a probability vector. As our results showed in AFONSO e Berton (2020), this conversion seemed to be essential for obtaining good results. Otherwise, the algorithm appeared to remove instances that are somewhat far away, even if surrounded by instances of the same class. However, this can make the gradient function quite messy. It can still be handled very well by optimization frameworks such as Tensorflow (ABADI et al., 2015). With that said, it's still interesting to consider the following relaxation of the problem, where we introduce a diagonal matrix $\mathbf{C}$:

$$C(\mathbf{C}, \mathbf{\Omega}) = \left\| \mathbf{C}\widetilde{\mathbf{P}}\mathbf{\Omega}\mathbf{Y}_{\mathcal{L}} - \mathbf{Y}_{\mathcal{L}} \right\|^2 \qquad \text{such that} \quad \mathbf{0}_{l\times l} \leq \mathbf{\Omega}, \mathbf{C} \tag{4.23}$$

By using specialized software (LAUE; MITTERREITER; GIESEN, 2018), we verified that the gradient corresponding to this is given by the diagonal entries of the following:

$$\frac{\partial C}{\partial \mathbf{\Omega}} = 2(\mathbf{C}\widetilde{\mathbf{P}})^{\top}\mathbf{C}\widetilde{\mathbf{P}}\mathbf{\Omega}\mathbf{Y}_{\mathcal{L}}\mathbf{Y}_{\mathcal{L}}^{\top} - 2\mathbf{Y}_{\mathcal{L}}\mathbf{Y}_{\mathcal{L}}^{\top} \tag{4.24}$$

$$\frac{\partial C}{\partial \mathbf{C}} = 2\left(\mathbf{C}(\widetilde{\mathbf{P}}\mathbf{\Omega}\mathbf{Y}_{\mathcal{L}}) - \mathbf{Y}_{\mathcal{L}}\right)(\widetilde{\mathbf{P}}\mathbf{\Omega}\mathbf{Y}_{\mathcal{L}})^{\top} \tag{4.25}$$

$\mathbf{Y}_{\mathcal{L}}\mathbf{Y}_{\mathcal{L}}^{\top}$ acts as a masking operator. In particular, $(\mathbf{Y}_{\mathcal{L}}\mathbf{Y}_{\mathcal{L}}^{\top})_{ij}$ is equal to 1 if instances i, j share the same label, and zero otherwise. For any matrix $\mathbf{A}$ and indices i, j, $(\mathbf{A}\mathbf{Y}_{\mathcal{L}}\mathbf{Y}_{\mathcal{L}}^{\top})_{ij}$ is equal to a special sum of the elements of $\mathbf{A}$'s $i-th$ row, where we only include the elements corresponding to the same class as instance j. In addition, $\mathbf{Y}_{\mathcal{L}}\mathbf{Y}_{\mathcal{L}}^{\top}$ is equal to one in the diagonal entries. Finally, if we use Lemma 2.4.2 to unravel $(\mathbf{C}\widetilde{\mathbf{P}})^{\top}\mathbf{C}\widetilde{\mathbf{P}}$, we can rewrite the partial derivative as:

$$\left(\frac{\partial C}{\partial \mathbf{\Omega}}\right)_{ii} = 2\left(-1 + \sum_{j:(\mathbf{Y}_{\mathcal{L}}\mathbf{Y}_{\mathcal{L}}^{\top})_{ij}=1} \mathbf{\Omega}_{jj}(\mathbf{C}\widetilde{\mathbf{P}})_{[:,i]} \cdot (\mathbf{C}\widetilde{\mathbf{P}})_{[:,j]}\right) \tag{4.26}$$

$$= 2\left(-1 + \sum_{j:(\mathbf{Y}_{\mathcal{L}}\mathbf{Y}_{\mathcal{L}}^{\top})_{ij}=1} \mathbf{\Omega}_{jj} \sum_{1\leq k\leq l} (\mathbf{C}_{kk}^2\widetilde{\mathbf{P}}_{ki}\widetilde{\mathbf{P}}_{kj})\right) \tag{4.27}$$

This means that, for a fixed $\mathbf{C}$, we can obtain the solution by solving a linear system. However, the $\mathbf{\Omega}$ and $\mathbf{C}$ variables appear to be tightly coupled. This also is true for the other partial

derivative:

$$(\frac{\partial C}{\partial \mathbf{C}})_{ii} = 2\left(\sum_j \mathbf{C}_{ii}(\mathbf{\Omega}_{jj}\widetilde{\mathbf{P}}_{ij})^2 - (\mathbf{Y}_{\mathcal{L}})_{ij}\mathbf{\Omega}_{jj}\widetilde{\mathbf{P}}_{ij}\right) \tag{4.28}$$

Therefore, to implement a proper solver, one can either use automated frameworks such as Tensorflow, or implement the optimization procedure manually, by making use of the derivatives we have provided. For future work, we aim to consider the properties of this "relaxed" problem more closely, including convexity analysis. In practice, it seemed to converge faster than the original formulation.

Because the number of iterations, or equivalently, the number of removed labels r is eliminated, we have decided to name our approach `LGC_LVO_Auto`: *the automatic leave-one-out filter based on the LGC algorithm*. The `LGC_LVO_Auto` solution is, in principle, unique for each α parameter of LGC. In practice, we must specify some number of iterations for our gradient descent optimizer, but this should not have an effect on the solution, and we can substitute it with any criteria for convergence.

4.2.5.1 An Addendum: Cross-entropy loss

While we were experimenting with our filter, we found out that utilizing **cross-entropy loss** yielded better results than **squared error loss**. The mean cross entropy loss is defined as

$$xent(\mathbf{Y}, \mathbf{F}) = \frac{1}{l}\sum_{1\leq i\leq l}\sum_{1\leq j\leq c} -\mathbf{F}_{ij}log(\mathbf{Y}_{ij}) \tag{4.29}$$

Minimizing cross-entropy also minimizes the **Kullback-Leibler divergence**, which appropriately measures the difference between distributions. This probabilistic interpretation is part of the reason that cross-entropy is usually chosen over mean squared error whenever we are dealing with a classification problem (LECUN; BENGIO; HINTON, 2015). The `LGC_LVO_Auto` cost turns into

$$\min_{\mathbf{\Omega}} xent(deg(\widetilde{\mathbf{P}}\mathbf{\Omega}\mathbf{Y}_{\mathcal{L}})^{-1}\widetilde{\mathbf{P}}\mathbf{\Omega}\mathbf{Y}_{\mathcal{L}}, \mathbf{Y}_{\mathcal{L}}) \qquad \text{such that} \quad \mathbf{0}_{l\times l} \leq \mathbf{\Omega} \tag{4.30}$$

4.2.5.2 Correction of labels: beyond filtering

One interesting thing to take into account is that our solution, in theory, could improve performance even if the labels are not noisy. Let us recall the spiral scenario, i.e. Figure 1. If we increased the dispersion of each spiral, we would have an overlapping region. If we sample that region, there is about a fifty-fifty chance of obtaining a label of each class. There was no mistake in the labelling process, but we should still value labels from these regions differently. As a matter of fact, degree-based label weights have been employed before (WANG; JEBARA; CHANG, 2008). Ideally, our optimization procedure could be able to learn such a rule.

4.2.5.3 Correction of labels: beyond LGC

Even though our filter was initially developed with LGC in mind, it is a very general method that can be readily extended to other GSSL classifiers, such as the generalized LapRLS (Section 2.5.4). After all, the only thing we need is a matrix relating labelled instances to one another. We can simply take the matrix

$$(\mathbf{K}(\mathbf{\Sigma K} + \lambda_A \mathbf{I_{n\times n}} + \lambda_I \mathbf{LK})^{-1})_{\mathcal{LL}} \tag{4.31}$$

The diagonal is set to zero, and we proceed as we normally would.

4.2.5.4 The limitations of ours and other SSL filters

We have already mentioned, in the introduction, that semi-supervised learning is seldom completely "safe". This is also true when we are detecting noisy labels. In Afonso e Berton (2020), we found out that Laplacian Eigenmaps (LE) to be very effective at avoiding overfitting to noisy labels. This is because of its underlying assumption.

Assumption 4.2.1 – *Smooth Eigenbasis assumption*
The data can be accurately described by just the smoothest eigenfunctions

The eigenfunctions of a graph are very closely related to graph cuts and unsupervised clustering. As a result, the LE approach works wonderfully when the data consists of two high-dimensional Gaussians, whereas LGC does poorly. On the other hand, LE does not perform very well for complex datasets such as COIL and MNIST. For `LGC_LVO_Auto`, the underlying assumption is:

Assumption 4.2.2 – *`LGC_LVO_Auto` Assumption*
If we were to exclude the noisy labels, the LGC random walk (or whichever unsupervised relation between instances) accurately models the data.

As a result, it would be naive to expect good results on datasets where LGC fails. The best we can hope for is that to consistently improve the LGC baseline.

4.3 Methodology

Next, we will go over the methods, tools and techniques that we have employed to achieve our goals.

4.3.1 Programming Language

We have chosen to use `Python 3` as our programming language for the implementation of this project. One distinct advantage of Python is the amount of packages available that simplify the pipeline of reading data, manipulating matrices and performing optimization. Some of the notable packages that we have used are:

- **numpy** for general arithmetic calculations
- **scipy** for its ability to store and work with sparse matrices.
- **faiss-gpu** to perform fast computation of the KNN graph for affinity matrix computation
- **tensorflow-gpu** for fast execution of critical inference methods. We use Tensorflow to implement the approximation of the propagation submatrix $\mathbf{P}_{\mathcal{L}}$, as well as for the automatic optimization of labels.

4.3.2 Experiment setup

Running experiments is a very time-consuming process. This is partly due to the necessity of repetition with different random seeds. We have found that some competing ℓ_1-norm methods only varied the noisy labels, while keeping the labeled samples the same throughout (GONG et al., 2017). We believe this to be insufficient, so we ensure that the random seed also controls the labeling process. In total, 20 random seeds are used that uniquely identify the correct and incorrect labels.

We have chosen to use the Noisy Completely at Random (NCAR) model. As established by (FRÉNAY; VERLEYSEN, 2014), datasets with identified noisy labels are extremely rare, therefore the most common approach is to inject NCAR noise artificially into the dataset. This statement was also validated by our literature review (GONG et al., 2017; LU et al., 2015; WANG; JEBARA; CHANG, 2008; WANG; JIANG; CHANG, 2009).

For the construction of affinity matrix, we used a **mutKNN** graph with $k = 15$ neighbours, with an RBF kernel (see Equation 2.16) whose σ was set heuristically as one third of the distance to the 10th neighbour, as in Chapelle, Schölkopf e Zien (2006). An exception to this rule was made for the `ISOLET` dataset, so that a more direct comparison with Gong et al. (2017) could be made: we use a $\sigma = 100$, with $k = 10$ neighbours. Such a direct comparison could not be made to the results in Lu et al. (2015), because it uses a special graph designed for scalability. To make up for this, we also tested the LGC baseline, and verified that it behaved very similarly to their implementation of LGC. For `LGC_LVO_Auto`, we minimized both loss functions (Equations 4.22 and 4.29) with the use of Tensorflow. The chosen optimized was Adam, with a learning rate of 0.7 and 5000 iterations. For the approximation of the propagation matrix, we used $t = 1000$ iterations throughout.

4.3.3 Datasets

In this section, we present the chosen datasets for our experiments. Besides the description of each dataset, we also use dimensionality reduction techniques in order to provide a clearer visualization. The two techniques we make use of (for visualization purposes only) are: **Locally Linear Embedding** (ROWEIS; SAUL, 2000) and **t-distributed stochastic neighbor embedding (t-SNE)** (MAATEN; HINTON, 2008). LLE is a dimensionality reduction technique that exploit the local linearity of the manifold, expressing each instance as a weighted linear combination of its neighbours. Then, it solves a sparse eigenvalue problem to find a low-dimensional representation that preserves those weights. In our case, a two-dimensional representation is used.

Digit1 The Digit1 dataset (CHAPELLE; SCHÖLKOPF; ZIEN, 2006) consists of artificially generated images of the digit 1. We can consider that it has the following degrees of freedom: two for translation, one for rotation, one for line thickness, one for the length of the small line at the bottom. Because of this intrinsic low-dimensionality, it can be considered as a low-dimensional submanifold of the input space, regardless of the resolution of the image. The task is made slightly harder by an obfuscation algorithm (CHAPELLE; SCHÖLKOPF; ZIEN, 2006), including the omission of certain pixels and downsampling, which results in instances with 241 attributes. According to (CHAPELLE; SCHÖLKOPF; ZIEN, 2006), this dataset does not show an obvious cluster structure. Figure 3 plots the **Locally Linear Embedding** (LLE) along with the edges corresponding to a symKNN graph with $k = 15$.

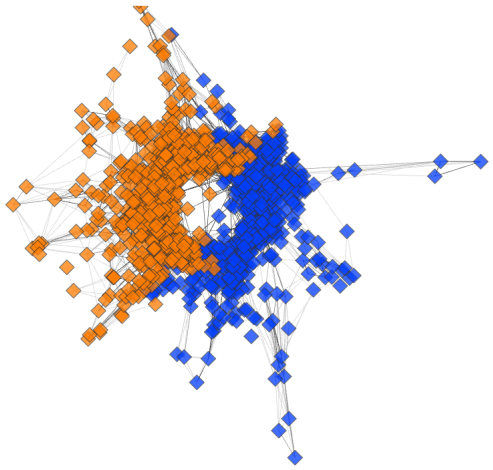


Figure 3: Locally linear embedding of the Digit1 dataset, based on linear reconstruction with $k = 15$ neighbours.

ISOLET The Isolated Letter Speech Recognition (COLE, 1990) is a dataset related to audio data. To create this dataset, 150 subjects had to pronounce each of the 26 letters of the alphabet

twice. The actual number of instances is 7797, due to 3 missing examples. There are 26 classes, which identify the correct letter being uttered within the range A-Z.

Due to ISOLET having a significant amount of classes, it is harder to visualize the distribution of classes with just a colour palette. In Figure 4, we use different shapes to identify each letter, in addition to its color. It is interesting to observe how some letters with a similar pronunciation are grouped together. Some examples would be: *F*, *S* and *X*; *U*, *Q* and *W*.

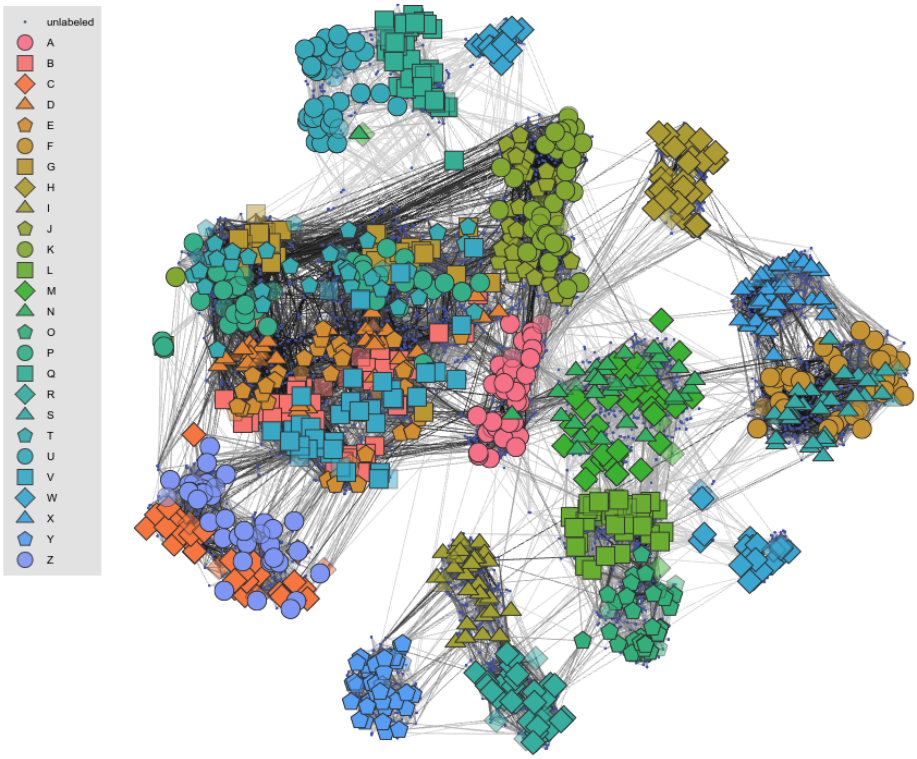


Figure 4: T-SNE embedding of ISOLET.

MNIST MNIST is a handwritten digit dataset comprising 70000 grayscale images, each with a height and width of 28 pixels, for a total input dimension of 784. Each digit from 0 to 9 is represented. The frequency of digits is nearly balanced, as each class has a number of representatives in the range of 6 to 8 thousand. This the dataset with most instances by a wide margin, with almost 47 times as many as as Digit1, and 9 times as many as ISOLET. LLE and t-SNE embeddings for this dataset are shown in Figure 5.

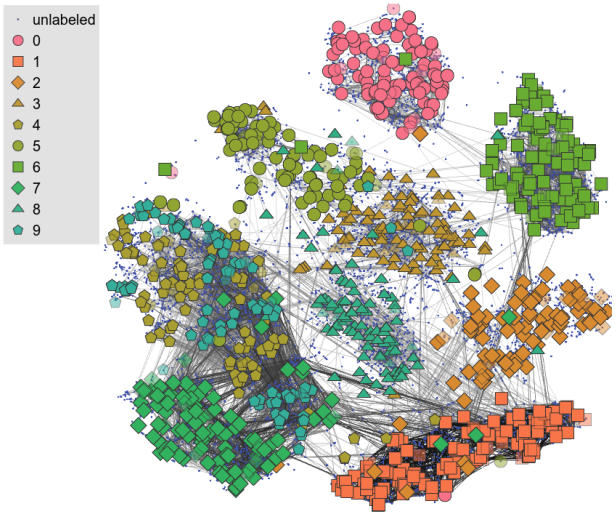


Figure 5: T-SNE embedding of a subset with 10000 instances from MNIST.

4.3.4 Workflow

Our basic workflow is essentially the one developed during the author's master degree. (AFONSO, 2020). We abstract the concept of an **experiment** as a list of **configurations**. Each configuration may have different parameters controlling each part of the execution. The interactions between the main components are illustrated in Figure 7, whereas Figure 6 shows the State Diagram for a single execution.

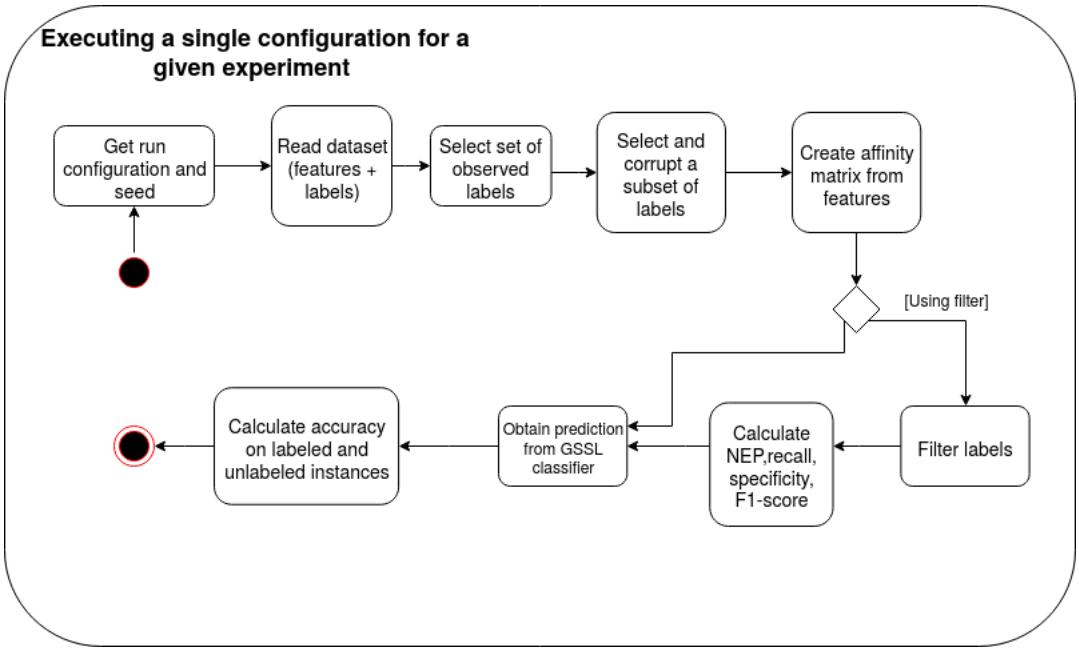


Figure 6: A state diagram, illustrating the flow of execution of a single configuration. Source: (AFONSO, 2020).

A typical configuration is executed as following: firstly, the necessary information is obtained from the **configuration specifier**. These include parameters about the dataset, labelling and noise processes, as well as parameters for filters and classifiers. As such, this information is forwarded to the corresponding components. The next step is to load the dataset, via the **dataset loader**. Next, the **noise generator** generates a set of noisy labels. The KNN graph is created by the **affinity matrix generator**, which also calculates the weights of the RBF kernel. The affinity matrix and noisy labels are given to the **filter** component, which outputs a third set of labels that corresponds to its attempt at fixing label noise. This set of labels is given to the **GSSL classifier**, as well as the affinity matrix. The output of the classifier is evaluated (by an **evaluator**), and the results are written to a CSV file. A special program is responsible for merging and calculating statistics about results corresponding to the same experiment. As shown in Figure 7, the GSSL classifier will receive the noisy labels if there is no filter, or the filtered labels otherwise.

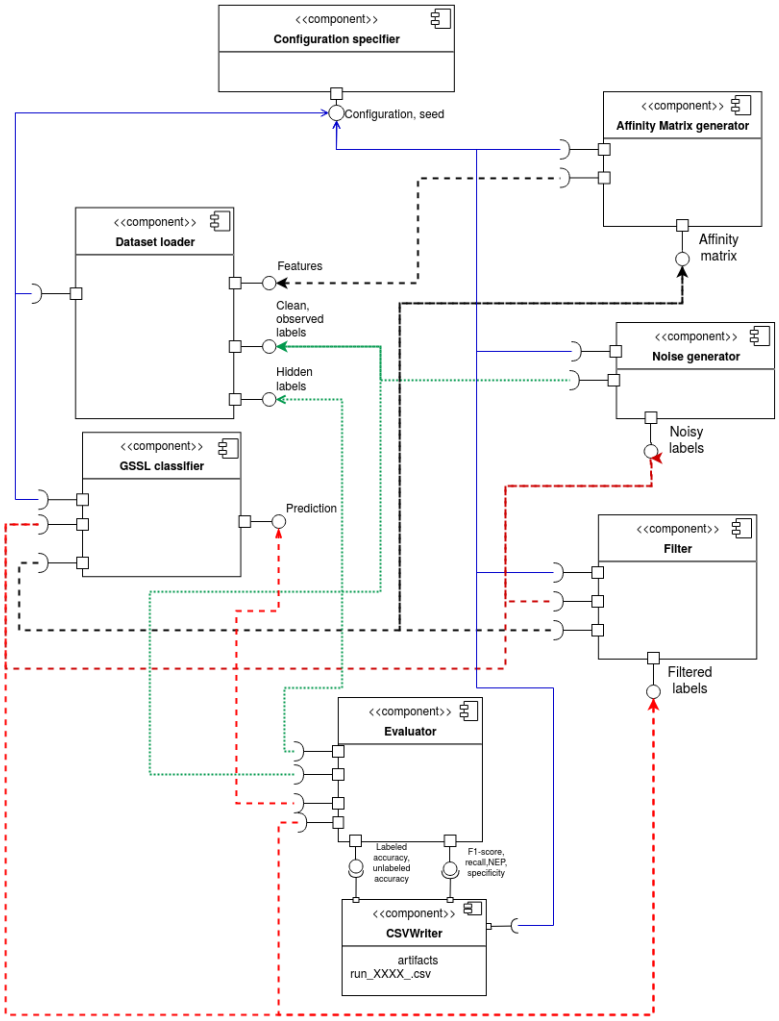


Figure 7: Diagram showing the interfaces required by each component of our system. Colour-coded for convenience. Blue: provides interface to the seed and value of hyper-parameters for the configuration. Green: provides interface to perfect label information (on labelled instances, and also unlabelled for evaluation purposes). Red: interface to imperfect label information. Black: any other interface. Source: (AFONSO, 2020).

5 Results

In this chapter, we will go over the results of the experiments. In the first experiment (Section 5.1), we measure up `LGC_LVO_Auto` against the LGC baseline and some ℓ_1 norm approaches, using the ISOLET dataset benchmark. The next experiment (Section 5.2) once again evaluates `LGC_LVO_Auto`, this time replicating the MNIST benchmark found in LSSC's paper. Finally, on Section 5.3 we analyze and discuss the behaviour related to our threshold approach, and complement it with the `LGC_LVO_Auto` results on Digit1 with few labels.

5.1 Experiment 1 (ISOLET): Vs SIIS, LSSC, GTF, GFHF, LGC

Experiment setting In this experiment, we compared `LGC_LVO_Auto` to the baselines reported in Gong et al. (2017), specifically for the ISOLET dataset. Unlike the authors, our 20 different seeds also control both the **label selection and noise processes**. The graph construction was performed exactly as in Gong et al. (2017), a 10-nearest neighbours graph with the width σ of the RBF kernel set to 100. We emphasize that the reported results by the authors correspond to the best-performing parameters, divided for each individual noise level. Recall the SIIS cost function, as given by Equation 3.9. In their work, λ_1 is tuned to 10^5, 10^2, 10^2 and 10^2 for each respective noise rate; λ_2 is kept to 10, and the number of eigenfunctions is $m = 30$. We could not find any implementation code for SIIS, so we had to manually reproduce it ourselves. As for parameter selection for `LGC_LVO_Auto`, we simply set $\alpha = 0.9$ (equivalently, $\mu = 0.1111$). We reiterate that **having a single parameter** is a **strength of our approach**. In future work, we will try to minimize leave-one-out error with respect to α or find some suitable heuristic.

Experiment results The results are contained in Tables 1a and 1b. With respect to the accuracy on **unlabelled examples**, we observed that:

- SIIS appeared to have a slight edge in the noiseless scenario.
- **LGC's own inherent robustness was evident**. When 60% noise was injected, it went from 84.72 to 70.69, a decrease of 16.55%. In comparison, SIIS had a decrease of 14.39%; GFHF a decrease of 22.08%; GTF a decrease of 21.82%.
- The cross-entropy version of `LGC_LVO_Auto`, again for 60% noise, had a decrease of 11.52%, so **`LGC_LVO_Auto`(xent) had the lowest percentual decrease of them all**.

- **`LGC_LVO_Auto`(MSE) disappointed** for both labelled and unlabelled instances. Except for the noiseless scenario, it was consistently outdone by `LGC_LVO_Auto`'s cross-entropy version.
- **`LGC_LVO_Auto` was not noticeably superior to LGC when there was less than 60% noise**. This is an unfortunate phenomenon that was also observed in AFONSO e Berton (2020), where we looked at an optimistic scenario for the filter.

With respect to the accuracy on **unlabelled examples**, we observed that:

- **LGC was unable to correct noisy labels**.
- `LGC_LVO_Auto`$(xent)$ discarded around 5% of the labels for the noiseless scenario, which is **better than SIIS and LSSC**.
- Moreover, **`LGC_LVO_Auto`**$(xent)$ had the **highest average accuracy on labelled instances** for $20\%, 40\%, 60\%$ label noise.

Remarks Overall, `LGC_LVO_Auto` was **very successful at the task of label diagnosis**: it was able to detect and remove labels with overall better performance than every ℓ_1-norm method. On the other hand, **this did not translate too well for unlabelled instance classification**. We believe that this could be because of the high proportion and quantity of labels, which is higher than most benchmarks found during a systematic review conducted during the author's master degree. On a more optimistic note, **`LGC_LVO_Auto` did not cause accuracy to decrease significantly in the noiseless scenario**.

(a) Accuracy on **unlabelled examples** only

Dataset	Noise Level	LSSC	GTF	GFHF	SIIS
	0%	84.8 ± 0.0	70.1 ± 0.0	86.5 ± 0.0	**85.4 ± 0.0**
ISOLET	20%	82.8 ± 0.3	69.9 ± 0.2	81.6 ± 0.4	**84.9 ± 0.6**
(1040/7797 labels)	40%	78.5 ± 0.6	59.8 ± 0.3	79.7 ± 1.0	**80.2 ± 1.3**
reported results	60 %	67.5 ± 1.8	54.8 ± 0.5	67.4 ± 1.5	**74.9 ± 1.4**
Dataset	Noise Level	LGC	`LGC_LVO_Auto` (MSE)	`LGC_LVO_Auto` (XENT)	SIIS
	0%	84.71 ± 0.56	84.21±0.4	84.22±0.45	**85.24 ± 0.32**
ISOLET	20%	82.89 ± 0.59	81.6±0.63	82.56±0.62	**83.69 ± 0.33**
(1040/7797 labels)	40%	79.33 ± 0.92	77.73±0.96	80.23±0.74	**80.88 ± 0.77**
our results	60%	70.69 ± 1.01	68.98±1.81	**74.51±1.75**	72.97 ± 1.16

(b) Accuracy on **labelled examples** after label correction

Dataset	Noise Level	LSSC	GTF	GFHF	SIIS
	0%	89.9 ± 0.0	95.8 ± 0.0	**100.00 ± 0.0**	91.1 ± 0.0
ISOLET	20%	87.7 ± 0.3	79.8 ± 0.7	80.00 ± 0.00	**90.5 ± 0.8**
(1040/7797 labels)	40%	82.9 ± 0.9	63.3 ± 0.4	60.00 ± 0.00	**83.6 ± 1.0**
reported results	60 %	71.8 ± 1.7	55.3 ± 0.6	40.00 ± 0.00	**77.4 ± 1.0**
Dataset	Noise Level	LGC	`LGC_LVO_Auto` (MSE)	`LGC_LVO_Auto` (XENT)	SIIS
	0%	**99.9 ± 0.02**	97.36±0.52	95.01±0.58	90.24 ± 0.69
ISOLET	20%	80.84 ± 0.27	90.93±1.28	**91.52±0.79**	88.5 ± 1.07
(1040/7797 labels)	40%	60.24 ± 0.20	82.54 ± 0.92	**87.19±0.89**	85.25 ± 0.96
our results	60%	40.00 ± 0.04	71.16 ± 1.79	**79.14±1.72**	76.34 ± 1.53

Table 1: Accuracy on `ISOLET` dataset

5.2 Experiment 2 (MNIST): Vs LSSC, Eigenfunction, LGC

Experiment setting This experiment was based on Lu et al. (2015), where a few classifiers were tested on the MNIST dataset subject to label noise. In that paper, the parameters for the graph were tuned to minimize cross-validation error. Moreover, an anchor graph was used, which is a large-scale solution. We did not use such a graph, as **our tensorflow iterative implementation of `LGC_LVO_Auto` was efficient enough to perform classification on MNIST in just a few seconds**. In the future, we will investigate how the choice of the graph affects the performance of `LGC_LVO_Auto`. We were also **unable to run SIIS on this dataset**, as numpy's eigenvector extraction method did not scale well to MNIST. As we also included the results for LGC (without anchor graph), it is interesting to observe that its accuracy decreases similarly to the previously reported results: the main difference is better performance for the noiseless scenario, which is to be expected (the anchor graph is an approximation).

Once again, we simply set $\alpha = 0.9$ for `LGC_LVO_Auto`. We used a symKNN matrix with $k = 15$ neighbours, and a heuristic sigma $\sigma = 423.57$ obtained by taking one third of the mean distance to the 10th neighbour (as in (CHAPELLE; SCHÖLKOPF; ZIEN, 2006)).

(a) Accuracy on **unlabelled examples** only

Dataset	Noise Level	LSSC*	Eigenfunction*	LGC* (anchor graph)
MNIST	0%	**93.1 ± 0.7**	73.8 ± 1.6	90.4 ± 0.7
(100/70000 labels)	15%	**91.1 ± 2.0**	68.6 ± 2.8	83.5 ± 1.6
reported results	30%	**89.0 ± 3.6**	61.9 ± 4.0	74.4 ± 2.8
Dataset	Noise Level	`LGC_LVO_Auto` (MSE)	`LGC_LVO_Auto` (XENT)	LGC
MNIST	0%	91.7 ± 0.7	92.69 ± 1.19	**93.09 ± 0.92**
(100/70000 labels)	15%	86.48 ± 2.59	**90.45 ± 2.13**	85.40 ± 1.66
our results	30%	81.33 ± 4.43	**84.46 ± 3.89**	74.58 ± 2.6

(b) Accuracy on **labelled examples** after label correction

Dataset	Noise Level	`LGC_LVO_Auto` (MSE)	`LGC_LVO_Auto` (XENT)	LGC
MNIST	0%	99.5 ± 0.59	98.05 ± 0.59	**100.00 ± 0.0**
(100/70000 labels)	15%	95.05 ± 2.01	**96.10 ± 1.3**	85.00 ± 0.0
our results	30%	85.55±4.88	**89.75 ± 4.06**	70.00 ± 0.0

Table 2: Accuracy on `MNIST` dataset

Experiment results The results are found in Tables 2b and 2a. With respect to the accuracy on **unlabelled examples**, we observed that:

- **`LGC_LVO_Auto` with cross-entropy improved the LGC baseline significantly on unlabelled instances**. For 30% label noise, mean accuracy increases **from 74.58% to 84.46%**.
- The **mean squared error loss is once again consistently inferior to cross-entropy when there is noise**. However, here it yielded slightly better results when compared to the LGC baseline.
- Though not directly comparable, `LGC_LVO_Auto` was not able to obtain better results than LSSC.

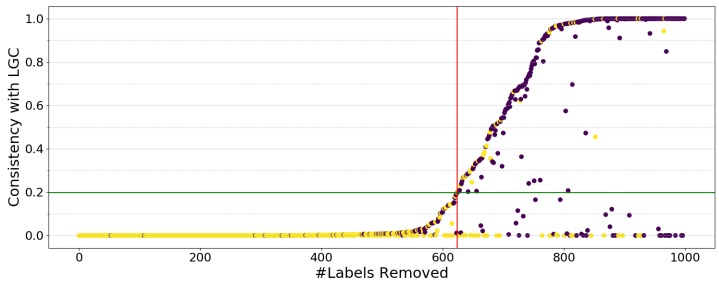
With respect to the accuracy on **labelled examples**, we observed that:

- **LGC was not able to correct the labelled instances**.
- **`LGC_LVO_Auto` with cross-entropy improved the LGC baseline significantly on labelled instances as well**. For 30% label noise, mean accuracy increases **from 70.00% to 89.75%**. This means that, on average, at least **2/3 noisy labels are fixed**.

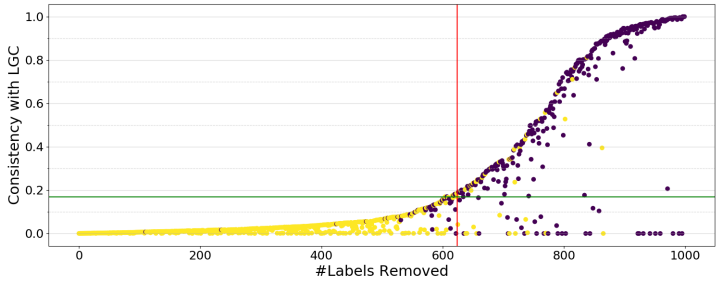
Remarks This experiment is a contrast to the previous one, where `LGC_LVO_Auto` was good for diagnosing noisy labels but not for classification. This time around, **performance is massively boosted for unlabelled instances as well**.

5.3 Experiment 3: Investigating the Optimal Threshold

For this experiment, we considered the **threshold approach** developed during this work, detailed in Section 2.3. When we set a threshold h, we are essentially telling our filter to stop when it first selects a label to be removed such that its consistency exceeds h.



(a) $\alpha = 0.1$ and 60% label noise



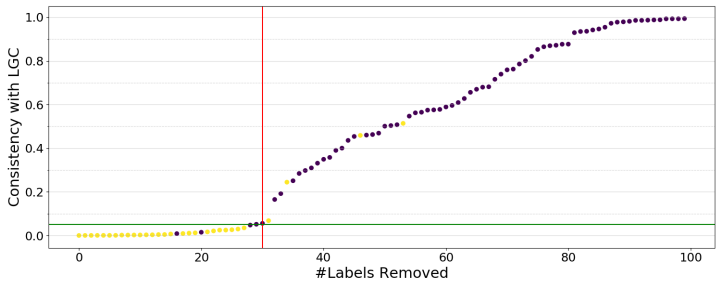
(b) $\alpha = 0.9$ and 60% label noise

Figure 8: Dynamic threshold plot for ISOLET

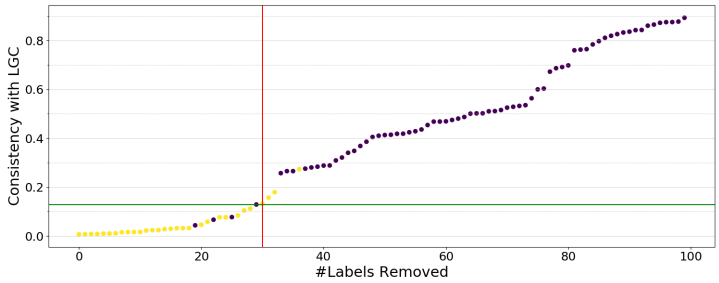
In this experiment, we generated a set of plots to evaluate if noisy labels are easily identifiable, and how the consistency measures changes depending on the random walk decay α. This can be seen, for example, in Figure 8a. There, the horizontal axis measures the number r of removed labels, whereas the vertical axis corresponds to the consistency value (Q value) of the label as it is being removed. Noisy labels are shown in yellow. This, of course, will not be the case when we present this plot to a user, but it is useful for this investigation because we

can show whether noisy labels are identified early and at which threshold. The red vertical line corresponds to the **total number of noisy labels**, and the green line the **corresponding threshold**. So, roughly, we should think of the green line as a threshold which is close to optimal. We can use either a **dynamic threshold plot**, where consistency (that is, matrix Q) is updated after each iteration of `LGC_LVOf`, or a **static threshold plot**, where we only consider the initial calculation of Q.

In Figure 8, we consider the ISOLET benchmark of Experiment 1. We can see that the majority of noisy labels are removed first, with a consistency value inferior to 0.2. Moreover, it is interesting to observe the general behaviour of each curve. They appear to roughly follow the form of a sigmoid function. The parameter setting $\alpha = 0.9$ corresponds to a smoother curve. This is expected, as a higher value of α distributes the contribution to each label, so that each label is "retrieved" by a higher amount of nearby labels. In addition, $\alpha = 0.1$ seemed to have lower precision, as it removed considerably more labels before, say, reaching 400 removed labels.



(a) $\alpha = 0.9$ and 30% label noise

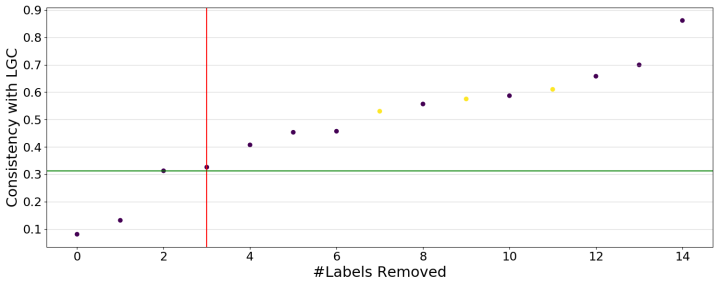


(b) $\alpha = 0.99$ and 30% label noise

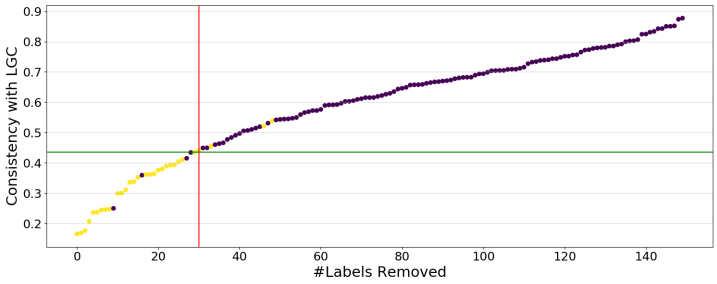
Figure 9: Static threshold plot for MNIST, 100/70000 labels

Figure 9 illustrates the static threshold plot for an instantiation of the MNIST benchmark with 30% label noise. In practice, we found the static threshold plot to be more intuitive, as it is monotonically non-decreasing. We also did not find a noticeable gap in quality between the order of instances detected in the static and dynamic setting. Therefore, it is possible that updating matrix Q during Algorithm 1 is not necessary after the first iteration. We remark that, for the MNIST problem with this specific random seed, $\alpha = 0.99$ appears to better allocate noisy labels to the leftmost cluster, before the first noticeable gap in consistency values.

Unfortunately, our filter seemed to not work so well with other datasets. As expected, it performs poorly when there are very few labels available. This is shown in Figure 10, where one can see that the noisy labels may actually have higher than average consistency. When more labels are available, this becomes less likely to happen, but there doesn't seem to be an obvious way to select this parameter. In practice, when there were 15 labelled instances out of 1500 in Digit1, the mean accuracy of `LGC_LVO_Auto`$(xent)$ was just barely improved upon the baseline, and this advantage is not good enough to make up for the increased variance in the prediction. In this scenario, we believe that regularizing by restricting the eigenfunction basis (smooth eigenbase pursuit) is a better option.



(a) 15 out of 15000 labels, $\alpha = 0.9$ and 20% label noise



(b) 150 out of 15000 labels, $\alpha = 0.9$ and 30% label noise

Figure 10: Identifiability issues in Digit1

Filter	Noise	Accuracy(unlabelled)
`LGC_LVO_Auto` (mse)	0%	88.83±4.9
`LGC_LVO_Auto` (xent)	0%	87.78±6.29
none	0%	**89.19±4.54**
`LGC_LVO_Auto` (mse)	10%	83.06±9
`LGC_LVO_Auto` (xent)	10%	**83.55±8.44**
none	10%	82.54±7.43
`LGC_LVO_Auto` (mse)	20%	75.84±12.09
`LGC_LVO_Auto` (xent)	20%	**77.04±11.4**
none	20%	75.91±8.81
`LGC_LVO_Auto` (mse)	35%	65.06±13.05
`LGC_LVO_Auto` (xent)	35%	**66.38±11.7**
none	35%	65.88±6.17

Filter	Noise	Accuracy(labelled)
`LGC_LVO_Auto` (mse)	0%	**99±3.18**
`LGC_LVO_Auto` (xent)	0%	97.67±5.28
none	0%	100±0
`LGC_LVO_Auto` (mse)	10%	93±5.36
`LGC_LVO_Auto` (xent)	10%	**93.67±6.49**
none	10%	93.33±0
`LGC_LVO_Auto` (mse)	20%	82±10.13
`LGC_LVO_Auto` (xent)	20%	**83.67±11.64**
none	20%	80±0
`LGC_LVO_Auto` (mse)	35%	67±13.58
`LGC_LVO_Auto` (xent)	35%	**69.33±12.72**
none	35%	66.67±0

Table 3: Accuracy for Digit1, $\alpha = 0.9$, 15/1500 labels.

6 Concluding remarks

In this work, we aimed to improve our recently developed filter `LGC_LVOf` in two ways: first, by introducing an intuitive threshold which is more intuitive than simply setting the amount of labels to be removed; secondly, via `LGC_LVO_Auto`, a novel approach that attempts to optimize labels without any supervision.

The development of this work has lead to the discovery of many different pros and cons that follow `LGC_LVOf` and `LGC_LVO_Auto`. Some of the advantages are:

1. We separate the solution $\mathbf{F} = \mathbf{PY}$, where $\mathbf{P}$ describes the expected relationship between labels.
2. By zeroing out the diagonal, we consider the classification for each individual label as if it was left out of the training set.
3. $\mathbf{P}$ is completely unsupervised. We can immediately evaluate the consistency of each $\mathbf{P}$ with our set of labels. This could be used for hypothesis selection.
4. Our approach can be readily extended to LapRLS, which generalizes LGC.
5. We only need to store an $l \times l$ submatrix, which makes it more feasible for larger datasets. If necessary, one could use an Anchor Graph (LIU; WANG; CHANG, 2012) to speed up the calculation of $\mathbf{P}$. In practice, the number of iterations t used to approximate $\mathbf{P}$ well enough for our purposes was very low (we set $t = 1000$).
6. `LGC_LVO_Auto` has only one parameter: α. In practice, $\alpha = 0.9$ seemed to work well for all our datasets.
7. The threshold approach for `LGC_LVOf` allows the user to be more conservative; the quantity related to this threshold has a very intuitive meaning: it measures how much the label is being contradicted by the other labels.
8. If we want a given subset of labels to be fully trusted, minimal changes to our frameworks are needed. For `LGC_LVO_Auto`, we simply zero out the gradient for those labels.

Unfortunately, there are some disadvantages as well:

1. Our current iteration of `LGC_LVO_Auto` is based on contradictions. Each label needs to be contradicted by some set of labels. As a result, when there are very few labels, `LGC_LVO_Auto` may become unstable. If there is only one label per class, it is entirely useless.

2. We do not yet make use of eigenfunctions, i.e. smooth eigenbase pursuit. This is one important cornerstone of GSSL approaches, and is complementary to ours. Our priority for future work is to integrate the prioritization of few eigenfunctions into this pipeline. Most smooth eigenbasis pursuit methods simply force you to choose a parameter m corresponding to the desired number of eigenfunctions. When this number is underestimated, classification performance drops significantly (AFONSO; BERTON, 2020). We would like to choose it automatically (by minimizing leave-one-out error or using some heuristic).

3. Our approach favours label redundancy. If there is an outlier in the data that cannot be explained by the model, it is expected to be removed.

As far as the results of our experiments, we found out that

- The cross-entropy loss version of `LGC_LVO_Auto` outperformed the mean squared error version, save for the noiseless scenario.
- For the benchmarks we considered, using this label tuning was not useful when there was no label noise.
- On the ISOLET benchmark, `LGC_LVO_Auto` was overall the superior approach at identifying noisy labels.
- In spite of diagnosing labels rather well, there was no added benefit in unlabelled acccuracy for ISOLET.
- On the other hand, massive improvements in accuracy on unlabelled data was obtained in the MNIST dataset. For 30% noise, it was increased by more than 10%.
- When using the threshold approach, noisy labels tend to accumulate below a given threshold.

For future work, we wish to incorporate smooth eigenbasis pursuit into `LGC_LVO_Auto`. We also aim to investigate why the higher percentage of corrected labels did not make a difference on ISOLET. This could be, e.g., because class proportions were altered. We will further investigate the optimization of LOO error within the family of hypothesis implicitly derived from the LGC formulation. We aim to investigate the use of anchor graphs or other large-scale techniques to improve the scalability of GTAM, so that it can be compared on these benchmarks.

Bibliography

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <http://tensorflow.org/>. Citado na página 66.

AFONSO, B. *Analysis of Label Noise in Graph-Based Semi-Supervised Learning*. Dissertação (Mestrado), 2020. Citado 9 vezes nas páginas 13, 26, 35, 51, 63, 64, 65, 72, and 73.

AFONSO, B.; BERTON, L. Analysis of label noise in graph-based semi-supervised learning. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. [S.l.: s.n.], 2020. p. 1127–1134. Citado 4 vezes nas páginas 26, 52, 68, and 82.

AFONSO, B.; BERTON, L. Identifying noisy labels with a transductive semi-supervised leave-one-out filter. *Pattern Recognition Letters*, 2020. ISSN 0167-8655. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167865520303603>. Citado 6 vezes nas páginas 26, 61, 64, 65, 66, and 76.

AFONSO, B. et al. Housing prices prediction with a deep learning and random forest ensemble. In: SBC. *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional*. [S.l.], 2019. p. 389–400. Citado na página 23.

AGRAWAL, R. et al. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In: ACM. *Proceedings of the 22nd international conference on World Wide Web*. [S.l.], 2013. p. 13–24. Citado na página 53.

ALPAYDIN, E. *Introduction to Machine Learning*. 2nd. ed. [S.l.]: The MIT Press, 2010. ISBN 026201243X. Citado 2 vezes nas páginas 23 and 29.

BELKIN, M.; NIYOGI, P. Using manifold structure for partially labeled classification. *Advances in Neural Information Processing Systems*, v. 15, p. 929–936, 2003. Citado na página 51.

BELKIN, M.; NIYOGI, P.; SINDHWANI, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, v. 7, n. Nov, p. 2399–2434, 2006. Citado 2 vezes nas páginas 45 and 47.

BERTON, L. *Construção de redes baseadas em vizinhança para o aprendizado semissupervisionado*. Tese (Doutorado) — Universidade de São Paulo, 2016. Citado na página 36.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: springer, 2006. Citado na página 32.

BREVE, F. A.; ZHAO, L.; QUILES, M. G. Semi-supervised learning from imperfect data through particle cooperation and competition. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2010. p. 1–8. Citado na página 54.

BREVE, F. A.; ZHAO, L.; QUILES, M. G. Particle competition and cooperation for semi-supervised learning with label noise. *Neurocomputing*, Elsevier, v. 160, p. 63–72, 2015. Citado na página 54.

CATUNDA, J. P. K.; SILVA, A. T. da; BERTON, L. Car plate character recognition via semi-supervised learning. In: IEEE. *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. [S.l.], 2019. p. 735–740. Citado na página 24.

CHANG, J. C.; AMERSHI, S.; KAMAR, E. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. [S.l.: s.n.], 2017. p. 2334–2346. Citado na página 25.

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. (Ed.). *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006. Disponível em: <http://www.kyb.tuebingen.mpg.de/ssl-book>. Citado 9 vezes nas páginas 24, 30, 31, 32, 33, 35, 69, 70, and 77.

CHEN, J. X. The evolution of computing: Alphago. *Computing in Science & Engineering*, IEEE Computer Society, v. 18, n. 4, p. 4–7, 2016. Citado na página 23.

COLE, R. The isolet spoken letter database. 1990. Citado na página 70.

CROWSTON, K. Amazon mechanical turk: A research tool for organizations and information systems scholars. In: *Shaping the Future of ICT Research. Methods and Approaches*. [S.l.]: Springer, 2012. p. 210–221. Citado na página 25.

DAUBECHIES, I. et al. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, Wiley Online Library, v. 63, n. 1, p. 1–38, 2010. Citado na página 54.

ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. *Machine Learning*, Springer, v. 109, n. 2, p. 373–440, 2020. Citado 2 vezes nas páginas 24 and 31.

FERGUS, R.; WEISS, Y.; TORRALBA, A. Semi-supervised learning in gigantic image collections. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2009. p. 522–530. Citado na página 53.

FRÉNAY, B.; VERLEYSEN, M. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, p. 845–869, 2014. Citado 3 vezes nas páginas 25, 33, and 69.

GONG, C. et al. Learning with inadequate and incorrect supervision. In: IEEE. *Data Mining (ICDM), 2017 IEEE International Conference on*. [S.l.], 2017. p. 889–894. Citado 3 vezes nas páginas 54, 69, and 75.

GREENE, D.; CUNNINGHAM, P. Practical solutions to the problem of diagonal dominance in kernel document clustering. In: *Proceedings of the 23rd international conference on Machine learning*. [S.l.: s.n.], 2006. p. 377–384. Citado na página 60.

GU, N.; FAN, M.; MENG, D. Robust semi-supervised classification for noisy labels based on self-paced learning. *IEEE Signal Processing Letters*, IEEE, v. 23, n. 12, p. 1806–1810, 2016. Citado na página 52.

HAMILTON, W. L. et al. Inducing domain-specific sentiment lexicons from unlabeled corpora. In: NIH PUBLIC ACCESS. *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*. [S.l.], 2016. v. 2016, p. 595. Citado na página 24.

HICKEY, R. J. Noise modelling and evaluating learning from examples. *Artificial Intelligence*, v. 82, n. 1, p. 157–179, 1996. Citado 2 vezes nas páginas 25 and 33.

HULL, J. J. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 16, n. 5, p. 550–554, 1994. Citado na página 53.

JOACHIMS, T. Transductive inference for text classification using support vector machines. In: *Icml*. [S.l.: s.n.], 1999. v. 99, p. 200–209. Citado na página 32.

KARP, R. M. Reducibility among combinatorial problems. In: *Complexity of computer computations*. [S.l.]: Springer, 1972. p. 85–103. Citado na página 56.

KRIJTHE, J. Robust semi-supervised learning: projections, limits & constraints. 2018. Citado 2 vezes nas páginas 24 and 31.

LAUE, S.; MITTERREITER, M.; GIESEN, J. Computing higher order derivatives of matrix and tensor expressions. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2018. Citado na página 66.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015. Citado 2 vezes nas páginas 23 and 67.

LEE, J. *Introduction to topological manifolds*. [S.l.]: Springer Science & Business Media, 2010. Citado na página 33.

LEWIS, D. D. et al. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, v. 5, n. Apr, p. 361–397, 2004. Citado na página 54.

LIU, W.; WANG, J.; CHANG, S.-F. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, IEEE, v. 100, n. 9, p. 2624–2638, 2012. Citado 2 vezes nas páginas 64 and 81.

LU, Z. et al. Noise-robust semi-supervised learning by large-scale sparse coding. In: *AAAI*. [S.l.: s.n.], 2015. p. 2828–2834. Citado 3 vezes nas páginas 53, 69, and 77.

MAATEN, L. v. d.; HINTON, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, v. 9, n. Nov, p. 2579–2605, 2008. Citado na página 70.

MANWANI, N.; SASTRY, P. Noise tolerance under risk minimization. *IEEE Transactions on Cybernetics*, IEEE, v. 43, n. 3, p. 1146–1151, 2013. Citado 2 vezes nas páginas 25 and 34.

MAO, Y. et al. Image classifier learning from noisy labels via generalized graph smoothness priors. In: IEEE. *Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), 2016 IEEE 12th*. [S.l.], 2016. p. 1–5. Citado na página 53.

MENG, D.; ZHAO, Q.; JIANG, L. What objective does self-paced learning indeed optimize? *arXiv preprint arXiv:1511.06049*, 2015. Citado na página 52.

MIT, C. *MIT center for biological and computation learning*. 2000. Disponível em: <http://www.ai.mit.edu/projects/cbcl>. Citado na página 53.

MITCHELL, T. *Machine Learning*. New York: McGraw-Hill, 1997. Citado 2 vezes nas páginas 23 and 29.

QUINLAN, J. R. Induction of decision trees. *Machine Learning*, v. 1, n. 1, p. 81–106, Mar 1986. Citado na página 33.

RIGGINS, F. J.; WAMBA, S. F. Research directions on the adoption, usage, and impact of the internet of things through the use of big data analytics. In: IEEE. *System Sciences (HICSS), 2015 48th Hawaii International Conference on*. [S.l.], 2015. p. 1531–1540. Citado na página 23.

ROWEIS, S. T.; SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, American Association for the Advancement of Science, v. 290, n. 5500, p. 2323–2326, 2000. Citado na página 70.

SAAD, Y.; SCHULTZ, M. H. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, SIAM, v. 7, n. 3, p. 856–869, 1986. Citado na página 55.

SAHA, B.; SRIVASTAVA, D. Data quality: The other face of big data. In: IEEE. *2014 IEEE 30th International Conference on Data Engineering*. [S.l.], 2014. p. 1294–1297. Citado na página 23.

SAMARIA, F. S.; HARTER, A. C. Parameterisation of a stochastic model for human face identification. In: IEEE. *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*. [S.l.], 1994. p. 138–142. Citado na página 53.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, IBM, v. 3, n. 3, p. 210–229, 1959. Citado na página 23.

SOLANO, A. et al. Smart vending machines in the era of internet of things. *Future Generation Computer Systems*, Elsevier, v. 76, p. 215–220, 2017. Citado na página 23.

SOUSA, C. A. R. d. *Constrained graph-based semi-supervised learning with higher order regularization*. Tese (Doutorado) — Universidade de São Paulo, 2017. Citado 6 vezes nas páginas 41, 46, 47, 49, 50, and 56.

SOUSA, S.; MILIOS, E.; BERTON, L. Word sense disambiguation: an evaluation study of semi-supervised approaches with word embeddings. In: *International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2020. p. 1–8. Citado na página 24.

SOUSA, S. B. da S.; DEL-FIACO, R. de C.; BERTON, L. Cluster analysis of homicide rates in the brazilian state of goiás from 2002 to 2014. In: IEEE. *2018 XLIV Latin American Computer Conference (CLEI)*. [S.l.], 2018. p. 445–454. Citado na página 24.

SZUMMER, M.; JAAKKOLA, T. Partially labeled classification with markov random walks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2002. p. 945–952. Citado na página 41.

TANG, J. et al. Image annotation by k nn-sparse graph-based label propagation over noisily tagged web images. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 2, n. 2, p. 14, 2011. Citado na página 55.

TENG, C.-M. *A Comparison of Noise Handling Techniques*. 2015. ArXiv preprint arXiv:511.06049. Citado na página 25.

WANG, J.; JEBARA, T.; CHANG, S.-F. Graph transduction via alternating minimization. In: ACM. *Proceedings of the 25th international conference on Machine learning*. [S.l.], 2008. p. 1144–1151. Citado 5 vezes nas páginas 26, 55, 56, 67, and 69.

WANG, J.; JEBARA, T.; CHANG, S.-F. Semi-supervised learning using greedy max-cut. *Journal of Machine Learning Research*, v. 14, n. Mar, p. 771–800, 2013. Citado 2 vezes nas páginas 57 and 58.

WANG, J.; JIANG, Y.-G.; CHANG, S.-F. Label diagnosis through self tuning for web image search. In: IEEE. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. [S.l.], 2009. p. 1390–1397. Citado 3 vezes nas páginas 26, 57, and 69.

WANG, Y.-X. et al. Trend filtering on graphs. *The Journal of Machine Learning Research*, JMLR. org, v. 17, n. 1, p. 3651–3691, 2016. Citado na página 53.

XIA, Z. et al. Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces. In: BIOMED CENTRAL. *BMC systems biology*. [S.l.], 2010. v. 4, n. 2, p. S6. Citado na página 24.

XIU, Y. et al. Multiple graph regularized graph transduction via greedy gradient max-cut. *Information Sciences*, Elsevier, v. 423, p. 187–199, 2018. Citado na página 26.

YOUNG, T. et al. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, IEEE, v. 13, n. 3, p. 55–75, 2018. Citado na página 23.

YUAN, D. et al. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*, 2016. Citado na página 24.

ZHOU, D. et al. Learning with local and global consistency. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2004. p. 321–328. Citado 4 vezes nas páginas 30, 42, 43, and 47.

ZHU; GHAHRAMANI, Z. Learning from labeled and unlabeled data with label propagation. *School Comput Sci Carnegie Mellon Univ Pittsburgh PA Tech Rep CMUCALD02107*, v. 54, n. CMU-CALD-02-107, p. 1–19, 2002. Citado 2 vezes nas páginas 26 and 41.

ZHU, X. *Semi-supervised learning literature survey*. [S.l.], 2005. Computer Sciences. Citado na página 24.

ZHU, X. *Semi-supervised learning with graphs*. Tese (Doutorado), 2005. Citado na página 25.

ZHU, X.; WU, X. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, v. 22, n. 3, p. 177–210, Nov 2004. Citado na página 33.